PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of

Masakazu ISOMURA

Application No.: 10/801,642

Filed: March 17, 2004 Docket No.: 119118

For: METHOD FOR REFERRING TO ADDRESS OF VECTOR DATA AND VECTOR PROCESSOR

CLAIM FOR PRIORITY

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

The benefit of the filing date of the following prior foreign application filed in the following foreign country is hereby requested for the above-identified patent application and the priority provided in 35 U.S.C. §119 is hereby claimed:

Japanese Patent Application No. 2003-203856 filed on July 30, 2003

In support of this claim, a certified copy of said original foreign application:

☒ is filed herewith.

It is requested that the file of this application be marked to indicate that the requirements of 35 U.S.C. §119 have been fulfilled and that the Patent and Trademark Office kindly acknowledge receipt of this document.

Respectfully submitted,

James A. Oliff
Registration No. 27,075

Thomas J. Pardini
Registration No. 30,411

JAO:TJP/mlo

Date: April 12, 2004

OLIFF & BERRIDGE, PLC
P.O. Box 19928
Alexandria, Virginia 22320
Telephone: (703) 836-6400

DEPOSIT ACCOUNT USE AUTHORIZATION
Please grant any extension necessary for entry;
Charge any fee due to our Deposit Account No. 15-0461

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年　７月３０日

出願番号
Application Number: 特願２００３－２０３８５６

[ST. 10/C]: [ＪＰ２００３－２０３８５６]

出願人
Applicant(s): セイコーエプソン株式会社

２００４年　４月　２日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号　出証特２００４－３０２７４９９

【書類名】　特許願
【整理番号】　J0100879
【あて先】　特許庁長官殿
【国際特許分類】　G06F 15/00
【発明者】
【住所又は居所】　長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社内
【氏名】　礒村　政一
【特許出願人】
【識別番号】　000002369
【氏名又は名称】　セイコーエプソン株式会社
【代理人】
【識別番号】　100095728
【弁理士】
【氏名又は名称】　上柳　雅誉
【連絡先】　０２６６－５２－３５２８
【選任した代理人】
【識別番号】　100107076
【弁理士】
【氏名又は名称】　藤綱　英吉
【選任した代理人】
【識別番号】　100107261
【弁理士】
【氏名又は名称】　須澤　修
【先の出願に基づく優先権主張】
【出願番号】　特願2003- 92381
【出願日】　平成15年 3月28日

【手数料の表示】

【予納台帳番号】　013044

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　0109826

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　ベクトルデータのアドレス参照方法およびベクトルプロセッサ

【特許請求の範囲】

【請求項１】　指標ベクトルを用いて、ベクトルデータの読み出しあるいは書き込みにおけるメモリアドレスの参照を行うアドレス参照方法であって、

指標ベクトルの要素を格納する要素格納用レジスタを複数領域に分割し、各領域に所定のコードを格納し、前記指標ベクトルの要素格納用レジスタそれぞれの所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能とすることを特徴とするベクトルデータのアドレス参照方法。

【請求項２】　前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレスに対する相対アドレスを示すコードを格納し、

前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコードと、該基準アドレスとに基づいて、参照先のメモリアドレスであるターゲットアドレスを算出することを特徴とする請求項１記載のベクトルデータのアドレス参照方法。

【請求項３】　前記指標ベクトルをベクトルレジスタに格納し、該ベクトルレジスタの各要素レジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴とする請求項２記載のベクトルデータのアドレス参照方法。

【請求項４】　前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレスに対する相対アドレスを示すコードを格納し、

前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコードと、該基準アドレスとに基づいて、参照先のメモリアドレスであるターゲットアドレスを算出し、算出されたターゲットアドレスを新たな参照基準アドレスとすることを特徴とする請求項１記載のベクトルデータのアドレス参照方法。

【請求項５】　前記相対アドレスを示すコードを前記要素格納用レジスタとしてのスカラーレジスタに格納し、該スカラーレジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴とする請求項４記載のベクト

ルデータのアドレス参照方法。

【請求項６】　前記ベクトルデータの読み出しあるいは書き込みに関するベクトル命令の実行中に、前記分割された各領域のうち、選択する領域を動的に変化させることを特徴とする請求項１～５のいずれかに記載のベクトルデータのアドレス参照方法。

【請求項７】　前記要素格納用レジスタの前記分割された各領域のうち、選択する領域を指定するための指定パターンを所定のレジスタに格納し、該指定パターンに基づいて前記分割された領域を指定することにより、所定の指標ベクトルを生成することを特徴とする請求項６記載のベクトルデータのアドレス参照方法。

【請求項８】　第１の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第１のレジスタと、第２の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第２のレジスタとを用意し、第１および第２のレジスタそれぞれにおける所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能とすることを特徴とする請求項１～７のいずれかに記載のベクトルデータのアドレス参照方法。

【請求項９】　指標ベクトルを用いて、ベクトルデータの読み出しあるいは書き込みにおけるメモリアドレスの参照を行うベクトルプロセッサであって、

指標ベクトルの要素を格納する要素格納用レジスタを備え、

該要素格納用レジスタは、複数領域に分割されて各領域に所定のコードが格納され、

該要素格納用レジスタそれぞれの所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能であることを特徴とするベクトルプロセッサ。

【請求項１０】　第１の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第１のレジスタと、第２の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第２のレジスタとを備え、第１および第２のレジスタそれぞれにおける所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能であることを特徴とする請求項９記載のベクトルプロセッサ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、ベクトルデータの読み出しあるいは書き込みにおけるアドレス参照方法およびベクトルプロセッサに関する。

【０００２】

【従来の技術】

従来、画像処理等において、不規則にメモリ上のデータの読み出しあるいは書き込むことが行われており、そのような処理を効率的に行うべく、種々の方法が提案されている。

例えば、画像処理において、１枚の画像データの中で、特定のブロックのデータをメモリからレジスタに格納する場合、ブロックの一行分の画素データをレジスタに格納した後、ブロックの次行の画素データを読み込むために、メモリ上の所定長離れたアドレスが参照される。

【０００３】

このように、複雑なアドレス参照を行うための方法として、参考文献「スーパーコンピュータ」（オーム社、長島重夫、田中義一著）に記載された技術が知られている。

本文献においては、多次元配列データをメモリからベクトルレジスタに読み込んだり、ベクトルレジスタからメモリに書き込んだりする場合に、間接指標ベクトル参照を用いる技術が開示されている。

【０００４】

間接指標ベクトル参照とは、メモリ上の参照するアドレス順序を格納したリスト（指標ベクトル）を用意しておき、そのリストを順に参照することによって、間接的にメモリ上の所定のアドレスを参照する方式である。

このような間接指標ベクトル参照を用いることにより、メモリ上の複雑なアドレス参照を行うことが可能となる。

【０００５】

【非特許文献１】

長島重夫、田中義一著「スーパーコンピュータ」オーム社、ｐ．３５

－４１

【０００６】

【発明が解決しようとする課題】

しかしながら、従来の技術においては、指標ベクトルをベクトルレジスタに格納しておく必要があるため、以下のような問題が生じていた。

第１に、指標ベクトルを用意しておく手順が必要となるため、プログラムのコードサイズや処理サイクル数が増加してしまう。

【０００７】

第２に、用意しておく指標ベクトルが増加すると、指標ベクトルを格納しておくベクトルレジスタも増加することから、本来の演算処理に用いるレジスタリソースが不足し、演算効率の低下を招いてしまう。

本発明の課題は、間接指標ベクトル参照をより効率的に行うことである。

【０００８】

【課題を解決するための手段】

以上の課題を解決するため、本発明は、

指標ベクトルを用いて、ベクトルデータの読み出しあるいは書き込みにおけるメモリアドレスの参照を行うアドレス参照方法であって、指標ベクトルの要素を格納する要素格納用レジスタ（例えば、図４のレジスタファイル４０におけるインデックスベクトルを格納しているレジスタ）を複数領域（例えば、上位および下位の領域）に分割し、各領域に所定のコードを格納し、前記指標ベクトルの要素格納用レジスタそれぞれの所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能とすることを特徴としている。

【０００９】

例えば、発明の実施の形態における命令コード内の“エクステンション”によって、インデックスレジスタにおいて分割されたいずれかの領域を選択する。そして、選択された領域のコードによって、指標ベクトル（インデックスベクトル）の要素が決定され、所定のメモリアドレスを指定する指標ベクトルが生成される。

【００１０】

また、前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレス（例えば、ベースアドレス）に対する相対アドレスを示すコードを格納し、前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコード（例えば、上位１６ビットあるいは下位１６ビット等）と、該基準アドレスとに基づいて、参照先のメモリアドレスであるターゲットアドレス（メモリ上の参照アドレス）を算出することを特徴としている。

【００１１】

この方法は、発明の実施の形態におけるインデックス修飾アドレッシングに関連する。

また、要素格納用レジスタにおける分割された領域のいずれを選択するかは、命令コード中で指定したり、選択する領域を指定する指定パターンをレジスタ等に記憶しておき、記憶された指定パターンを入力して領域を指定するといったことが可能である。

【００１２】

また、前記指標ベクトルをベクトルレジスタに格納し、該ベクトルレジスタの各要素レジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴としている。

また、前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレスに対する相対アドレスを示すコードを格納し、前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコードと、該基準アドレスとに基づいて、参照先のメモリアドレスであるターゲットアドレスを算出し、算出されたターゲットアドレスを新たな参照基準アドレスとすることを特徴としている。

【００１３】

この方法は、発明の実施の形態におけるポスト・レジスタ・アップデート・アドレッシングに関連する。

また、前記相対アドレスを示すコードを前記要素格納用レジスタとしてのスカラーレジスタに格納し、該スカラーレジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴としている。

【００１４】

また、前記ベクトルデータの読み出しあるいは書き込みに関するベクトル命令の実行中に、前記分割された各領域のうち、選択する領域を動的に変化させることを特徴としている。

また、前記要素格納用レジスタの前記分割された各領域のうち、選択する領域を指定するための指定パターンを所定のレジスタに格納し、該指定パターンに基づいて前記分割された領域を指定することにより、所定の指標ベクトルを生成することを特徴としている。

【００１５】

また、第１の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第１のレジスタ(例えば、発明の実施の形態中の「インデックスレジスタ」)と、第２の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第２のレジスタ（例えば、図９のサブインデックスレジスタ１４１，１４２）とを用意し、第１および第２のレジスタそれぞれにおける所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能とすることを特徴としている。

【００１６】

また、本発明は、

指標ベクトルを用いて、ベクトルデータの読み出しあるいは書き込みにおけるメモリアドレスの参照を行うベクトルプロセッサであって、指標ベクトルの要素を格納する要素格納用レジスタ（例えば、発明の実施の形態中の「インデックスレジスタ」の要素レジスタ）を備え、該要素格納用レジスタは、複数領域に分割されて各領域に所定のコードが格納され、該要素格納用レジスタそれぞれの所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能であることを特徴としている。

【００１７】

また、第１の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第１のレジスタ（例えば、発明の実施の形態中の「インデックスレジスタ」）と、第２の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第２のレジスタ（例えば、図９のサブインデックスレジスタ１４１，１４２）とを備え

、第１および第２のレジスタそれぞれにおける所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能であることを特徴としている。

【００１８】

ここで、第１のレジスタおよび第２のレジスタの要素格納用レジスタにおける分割された領域のいずれを選択するかは、命令コード中で指定したり、選択する領域を指定する指定パターンをレジスタ等に記憶しておき、記憶された指定パターンを入力して領域を指定するといったことが可能である。

本発明によれば、ベクトルレジスタの要素レジスタあるいはスカラーレジスタを複数の領域に分割して、それぞれの領域に指標ベクトルの要素となる所定のコードを格納する。そして、分割した領域のいずれかを選択し、その領域に格納されたコードを用いて、所定の指標ベクトルを取得することとしている。

【００１９】

したがって、１つの指標ベクトルを格納するレジスタの領域に、実質的に複数のインデックスベクトルを格納できることとなり、レジスタリソースを効率的に使用することが可能となる。

【００２０】

【発明の実施の形態】

以下、図を参照して本発明に係るベクトルプロセッサの実施の形態を説明する。

（第１の実施の形態）

本発明に係るベクトルプロセッサは、指標ベクトルを格納するベクトルレジスタあるいはスカラーレジスタにおいて、ベクトルレジスタの要素レジスタあるいはスカラーレジスタを分割して使用し、アドレッシング機能を拡張している。

【００２１】

したがって、初めに、このような機能を実現するための基本となる考え方について説明する。なお、本発明は、ベクトルプロセッサにおけるロード命令およびストア命令に関するものであるため、これらを中心に説明する。

ベクトルプロセッサにおいて、ロード命令あるいはストア命令のコードタイプには、アドレッシングモード（アドレス参照の方法）に応じて、以下の３種類が

規定されている。

【００２２】

図１は、ロード命令あるいはストア命令のフォーマットを示す図であり、（ａ）は、ベース相対アドレッシングに対応するＬＳ０タイプ、（ｂ）は、ポスト・オフセット・アップデート・アドレッシングに対応するＬＳ１タイプ、（ｃ）は、インデックス修飾アドレッシングおよびポスト・レジスタ・アップデート・アドレッシングに対応するＬＳ２タイプを示している。

【００２３】

図１において、ＬＳ０タイプはスカラーデータのロード命令およびストア命令に対応しており、ＬＳ１タイプおよびＬＳ２タイプは、スカラーデータおよびベクトルデータのロード命令およびストア命令に対応している。

本発明は、ベクトルデータのロード命令およびストア命令を取り扱うものであり、インデックス修飾アドレッシングおよびポスト・レジスタ・アップデート・アドレッシングに関するものであるため、ＬＳ２タイプについて説明する。

【００２４】

図１（ｃ）において、ＬＳ２タイプのフォーマットには、オペコード（opecode）、デスティネーション（dst）、ベース（base）、リピートアマウント（rptamt）、エクステンション（extension）、インデックス（index）の６つのフィールドが含まれている。

オペコードは、命令の内容を示すフィールドであり、ロード命令あるいはストア命令のいずれかを示すコードが含まれている。

【００２５】

デスティネーションは、ロードあるいはストアの対象となるデータが格納されたレジスタのアドレスを指定するフィールドである。即ち、ロード命令の場合、メモリから読み出したデータを書き込むレジスタのアドレスであり、ストア命令の場合、メモリに書き込むデータを読み出すレジスタのアドレスである。

ベースは、基準となるメモリアドレスが格納されたレジスタのアドレスを指定するフィールドである。

【００２６】

リピートアマウントは、ベクトル命令における命令の繰り返し回数（要素データ数）を示すフィールドであり、命令繰り返し回数が“１”である場合はスカラー命令、“１”以外である場合はベクトル命令となる。

エクステンションは、機能拡張用に用意された５ビットのフィールドであり、本発明においては、後述のように、エクステンションを利用して、分割した要素レジスタのアドレスを指定する。

【００２７】

インデックスは、ベースに示されたアドレスに対し、どのような修飾を施すか、即ち、ベースに示されたアドレスから参照するアドレス順序を示すフィールドである。本発明においては、後述のように、インデックスを利用して、アドレス修飾の拡張仕様を規定する。

なお、インデックスは、上述の指標ベクトルが格納されたベクトルレジスタあるいはスカラーレジスタ（以下、指標ベクトルが格納されたレジスタを「インデックスレジスタ」と言う。）を指定しており、インデックスを参照することによって、間接指標ベクトル参照が行われる。

【００２８】

続いて、本発明におけるアドレス修飾の拡張仕様について具体的に説明する。

図２は、インデックス修飾アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。なお、図２においては、アドレス修飾の拡張仕様を説明するため、Ｃ言語に準じた記述スタイルのプログラミング例を併せて示しており、後述の図３においても同様である。

【００２９】

図２において、エクステンションが“０００”である場合、インデックスとして指定されたレジスタ（インデックスレジスタ）の内容（指標ベクトル）を符号付きで解釈し、アドレス修飾を行う。

具体的には、ベースに示されたレジスタの値を常に基準とし、その値にインデックスに示されるアドレスを順次加算することにより、アドレス修飾を行う。

【００３０】

また、エクステンションが“００１”である場合、インデックスとして指定されたレジスタの下位１６ビットを符号付きで解釈し、アドレス修飾を行う。

また、エクステンションが“０１０”である場合、インデックスとして指定されたレジスタの上位１６ビットを符号付きで解釈し、アドレス修飾を行う。

さらに、エクステンションが“０１１”である場合、インデックスとして指定されたレジスタの下位１６ビットおよび上位１６ビットをそれぞれ符号付きで解釈し、交互にアドレス修飾を行う。具体的には、ベクトル命令における繰り返しの実行順序が偶数である場合、下位１６ビットによってアドレス修飾を行い、奇数である場合、上位１６ビットによってアドレス修飾を行う。

【００３１】

次に、ポスト・レジスタ・アップデート・アドレッシングの拡張仕様について説明する。

図３は、ポスト・レジスタ・アップデート・アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

【００３２】

図３において、エクステンションが“０００”である場合、インデックスとして指定されたレジスタの内容を符号付きで解釈し、アドレスの更新を行う。

具体的には、ベースに指定されたレジスタの値を所定のレジスタ（図３中のパラメータＴの値を格納するレジスタ）に一旦格納し、そのレジスタの値を更新しながらアドレス修飾を行う。

【００３３】

また、エクステンションが“００１”である場合、インデックスとして指定されたレジスタの下位１６ビットを符号付きで解釈し、アドレスの更新を行う。

また、エクステンションが“０１０”である場合、インデックスとして指定されたレジスタの上位１６ビットを符号付きで解釈し、アドレスの更新を行う。

さらに、エクステンションが“０１１”である場合、インデックスとして指定されたレジスタの下位１６ビットおよび上位１６ビットをそれぞれ符号付きで解釈し、交互にアドレスの更新を行う。

【００３４】

ここで、エクステンションが“０００”～“０１１”までの場合、ベクトル命令における命令の繰り返し回数の間、インデックスとして指定された内容に応じてアドレス修飾が行われ、繰り返し回数が終了した後、ベースに指定されたレジスタの値が更新される。

したがって、複数のロード命令あるいはストア命令を用いて、繰り返しロードあるいはストアを行う場合、ベースに指定されたレジスタが自動的に更新されるので、命令発行の都度、ベースのアドレスを別の命令で更新する必要がなく、直ちに命令の実行を行うことが可能である。

【００３５】

また、図３において、エクステンションが“１００”～“１１１”の場合、エクステンションが“０００”～“０１１”の場合とそれぞれ同様にアドレスの更新を行う。ただし、エクステンションが“１００”～“１１１”の場合、命令の繰り返し回数が終了した後、ベースに指定されたレジスタの値を更新することなく命令の実行を終了する。

【００３６】

次に、本実施の形態に係るベクトルプロセッサの構成を説明する。

図４は、本実施の形態に係るベクトルプロセッサ１の構成を示す図である。

図４において、ベクトルプロセッサ１は、メモリ１０と、メモリ制御部２０と、命令フェッチ部３０と、レジスタファイル４０と、ロードユニット５０と、ストアユニット６０と、演算ユニット７０とを含んで構成される。

【００３７】

メモリ１０は、ベクトルプロセッサ１に与えられる命令コードおよび演算対象となるデータを記憶している。

メモリ制御部２０は、メモリ１０に対するアクセス、即ち、データの読み出しや書き込みを制御する。例えば、メモリ制御部２０は、ロードユニット５０あるいはストアユニット６０によって指定されたメモリ１０のアドレスからデータを読み出したり、メモリ１０から読み出されたデータをレジスタファイル４０に出力したりする。

【００３８】

命令フェッチ部３０は、メモリ制御部２０を介して、メモリ１０から命令コードをフェッチし、一時的に記憶する。

レジスタファイル４０は、３２個のスカラーレジスタＳＲ０～ＳＲ３１と、８個の要素レジスタからなる８個のベクトルレジスタＶＲ０～ＶＲ７とを含んで構成され、メモリ１０から読み出されたデータおよび演算結果を一時的に記憶する。なお、以下の説明において、ベクトルレジスタの要素レジスタおよびスカラーレジスタは、３２ビットの幅を持つものとして説明する。

【００３９】

ロードユニット５０は、命令フェッチ部３０に記憶された命令コードがロード命令である場合に、メモリ１０から命令コードあるいはデータを読み出す処理を行う。

ストアユニット６０は、命令フェッチ部３０に記憶された命令コードがストア命令である場合に、メモリ１０にデータを書き込む処理を行う。

【００４０】

演算ユニット７０は、命令フェッチ部３０に記憶された命令コードが所定の演算命令である場合に、レジスタファイル４０に記憶された所定データを対象として演算処理を行う。

ここで、ロードユニット５０の構成について、詳細に説明する。

図５は、ロードユニット５０の内部構成を示すブロック図である。

【００４１】

図５において、ロードユニット５０は、命令パイプライン制御部５１と、インデックスレジスタ決定回路５２と、デスティネーションレジスタ決定回路５３と、アドレス演算回路５４と、パイプラインレジスタ（ＰＲ）５５，５６と、レジスタ５７とを含んで構成される。

命令パイプライン制御部５１は、ロードユニット５０全体を制御するものである。

【００４２】

インデックスレジスタ決定回路５２は、命令コードのインデックスフィールド

に基づいて、指標ベクトルが格納されたインデックスレジスタを選択する信号（インデックスレジスタ選択信号）を生成する。

なお、インデックス修飾アドレッシングの場合、基準として指定されたアドレス（ベースアドレス）を保持し、そのアドレスに対する各要素の相対アドレスを指定することで参照アドレスを生成する。即ち、インデックス修飾アドレッシングの場合、要素ごとに相対アドレスを指定するため、インデックスレジスタとしてベクトルレジスタが指定される。

【００４３】

一方、ポスト・レジスタ・アップデート・アドレッシングの場合、基準として指定されたアドレス（ベースアドレス）を参照アドレスとすると共に、そのアドレスに対する相対アドレスを指定してベースアドレスが更新される。そして、その更新されたベースアドレスに対する、次の要素の相対アドレスを指定することを繰り返して、参照アドレスを生成する。そのため、ポスト・レジスタ・アップデート・アドレッシングの場合、インデックスレジスタとしてスカラーレジスタとベクトルレジスタの双方が指定可能である。

【００４４】

デスティネーションレジスタ決定回路５３は、命令コードのデスティネーションフィールドおよびリピートアマウントフィールドに基づいて、デスティネーションアドレスを格納するためのデスティネーションレジスタを選択する信号（デスティネーションレジスタ選択信号）を生成する。

アドレス演算回路５４は、命令パイプライン制御部５１の指示に基づいて、レジスタファイル４０から入力されるベースアドレスおよびインデックスアドレス（インデックスによって指定されるアドレス）から、ロード命令の対象となるメモリ１０上のアドレス（ロードアドレス）を算出する。

【００４５】

ＰＲ５５，５６は、命令フェッチ部３０から入力されるデスティネーションフィールドおよびリピートアマウントフィールドのコードを一時的に記憶し、パイプライン処理において１サイクル遅延させて、デスティネーションレジスタ決定回路５３に出力する。

レジスタ５７は、命令フェッチ部３０から入力されるベースフィールドのコードを一時的に記憶する。

【００４６】

ここで、図５におけるアドレス演算回路５４の構成について説明する。

図６は、アドレス演算回路５４の構成例を示す図である。

図６において、アドレス演算回路５４は、Ｉレジスタ５４ａと、Ｔレジスタ５４ｂと、マルチプレクサ（ＭＵＸ）５４ｃ～５４ｅと、加算器５４ｆとを含んで構成される。

【００４７】

Ｉレジスタ５４ａは、レジスタファイル４０から入力されるインデックスアドレスを一時的に記憶する。

Ｔレジスタ５４ｂは、ＭＵＸ５４ｃを介してレジスタファイル４０から入力されるベースアドレスを一時的に記憶する。

ＭＵＸ５４ｃは、レジスタファイル４０から入力されるベースアドレスと加算器５４ｆの出力であるアップデート・ベースアドレスとのいずれかを切り替えて、Ｔレジスタ５４ｂに出力する。

【００４８】

ＭＵＸ５４ｄは、Ｉレジスタ５４ａから入力されるインデックスアドレスの上位あるいは下位のアドレスのいずれかを選択して加算器５４ｆに出力する。

ＭＵＸ５４ｅは、Ｔレジスタ５４ｂから入力されるベースアドレスと加算器５４ｆの出力であるアップデート・ベースアドレスとのいずれかを切り替えて、ベクトル命令の各繰り返しにおいて、ロード命令を実行するターゲットアドレスとして出力する。

【００４９】

加算器５４ｆは、Ｔレジスタ５４ｂから入力されるベースアドレスの値と、ＭＵＸ５４ｄから入力されるインデックスアドレスの上位あるいは下位のデータに基づくアドレスとを加算し、アップデート・ベースアドレス（更新されたベースアドレスの値）として出力する。

なお、図６において、ＭＵＸ５４ｃ～５４ｅは命令パイプライン制御部５１に

よって制御される。即ち、ＭＵＸ５４ｃ～５４ｅには、入力されるデータのいずれかを選択するための信号（選択指示信号）が命令パイプライン制御部５１から入力される。

【００５０】

また、アドレス演算回路５４は、インデックス修飾アドレッシングの場合およびポスト・アップデート・アドレッシングの場合それぞれに対応して、所定の動作を行う。

次に、動作を説明する。

初めに、図４を参照して、ベクトルプロセッサ１全体の動作について説明する。

【００５１】

ベクトルプロセッサ１において処理が行われる場合、メモリ制御部２０を介してメモリ１０から命令フェッチ部３０に命令コードが読み出される。

そして、ロードユニット５０、ストアユニット６０および演算ユニット７０のそれぞれに、命令フェッチ部３０から命令コードが出力される。

命令コードが入力されたロードユニット５０、ストアユニット６０および演算ユニット７０は、その命令コードをデコードし、それぞれのユニットに対応する命令である場合にのみ、命令を実行する。

【００５２】

ここでは、図５を参照しつつ、命令コードがロード命令である場合について説明する。

命令コードのオペコードがロード命令を示している場合（より詳細にはprefixコードが“０００”である場合）、ロードユニット５０が動作する。

まず、ロードユニット５０は、命令フェッチ部３０から受け取ったベースフィールドのコードを、ベースレジスタ・リード選択信号（データの読み出し時にベースレジスタを選択する信号）としてレジスタファイル４０に出力する。ベースレジスタ・リード選択信号は、レジスタファイル４０内のスカラーレジスタＳＲ０～ＳＲ３１のいずれかをベースレジスタとして選択するための信号である。

【００５３】

そして、ベースレジスタ・リード選択信号を受け取ったレジスタファイル４０から、ベースレジスタ・リード選択信号によって指定されたレジスタに記憶されたベースアドレスの値がロードユニット５０に入力される。

レジスタファイル４０から入力されたベースアドレスの値は、アドレス演算回路５４に入力される。

【００５４】

また、インデックスレジスタ決定回路５２は、命令フェッチ部３０から入力されたインデックスフィールドのコードおよびリピートアマウントフィールドのコードを受け取る。そして、インデックスレジスタ決定回路５２は、命令パイプライン制御部５１からの指示に従って、命令コードがベクトル命令であるかスカラー命令であるかを判定し、ベクトル命令である場合、インデックスレジスタ選択信号を、リピートアマウントに示される要素データ数分、レジスタファイル４０に順次出力する。このとき、インデックスレジスタ選択信号によって指定されたレジスタがベクトルレジスタである場合、指定されたベクトルレジスタ内の各要素レジスタを特定するための所定の選択信号が出力される。

【００５５】

インデックスレジスタ選択信号が入力されたレジスタファイル４０からは、インデックスレジスタ選択信号に示されるレジスタから、インデックスアドレスの値が、アドレス演算回路５４に順次入力される。

インデックスアドレスの値を受け取ったアドレス演算回路５４は、ベースアドレスの値と、インデックスアドレスの値とからロードアドレスを算出し、メモリ制御部２０に出力する。なお、アドレス演算回路５４の動作については後述する。

【００５６】

また、命令フェッチ部３０から入力されたデスティネーションフィールドのコードおよびリピートアマウントフィールドのコードは、パイプライン処理におけるタイミングを合わせるべく、ＰＲ５５，５６に一旦記憶された後、デスティネーションレジスタ決定回路５３に入力される。

すると、デスティネーションレジスタ決定回路５３は、デスティネーションレ

ジスタ選択信号を、メモリ１０からロードされたデータと同期させてレジスタファイル４０に出力する。

【００５７】

そして、レジスタファイル４０において、メモリ１０からロードされたデータが、所定のデスティネーションレジスタに順次記憶される。

さらに、ポスト・レジスタ・アップデート・アドレッシングの場合、アドレス演算回路５４によって出力されるアップデート・ベースアドレスをベースレジスタに書き込む。

【００５８】

したがって、ベースフィールドのコードをレジスタ５７に保持しておき、レジスタ５７によって出力されるコードをベースレジスタ・ライト選択信号（データの書き込み時にベースレジスタを選択する信号）として用い、ベースレジスタ・ライト信号（ベースレジスタに対する書き込み指示信号）が入力されることに対応して、ベースレジスタのデータが更新される。

【００５９】

次に、アドレス演算回路５４の動作について説明する。

まず、インデックス修飾アドレッシングの場合について説明する。

インデックス修飾アドレッシングの場合、サイクル“１”において、ベースアドレスの値が、ＭＵＸ５４ｃを介してＴレジスタ５４ｂに格納される。

一方、インデックスアドレスの値は、Ｉレジスタ５４ａに格納される。

【００６０】

次いで、サイクル“２”において、Ｔレジスタ５４ｂのベースアドレスの値と、Ｉレジスタ５４ａの上位あるいは下位半分のいずれかのアドレスの値とが、加算器５４ｆによって加算され、ＭＵＸ５４ｅを介してターゲットアドレス（ロードアドレス）として出力される。

そして、第３サイクル以後は、ベースアドレスの値がＴレジスタ５４ｂに保持された状態のまま、新たにインデックスアドレスが入力され、加算器５４ｆによって、ベースアドレスとの加算が順次行われる。

【００６１】

このとき、エクステンションフィールドのコードが入力された命令パイプライン制御部５１が、ＭＵＸ５４ｄを制御することにより、図２に示すインデックス修飾アドレッシングの拡張仕様が実現される。

即ち、エクステンションが“０００”である場合、Ｉレジスタ５４ａに格納されたデータ（ここでは３２ビット）が、そのまま加算器５４ｆに入力される。

【００６２】

一方、エクステンションが“００１”である場合、Ｉレジスタ５４ａの下位１６ビットを符号拡張し、３２ビットとされたデータが加算器５４ｆに入力される。

また、エクステンションが“０１０”である場合、Ｉレジスタ５４ａの上位１６ビットを符号拡張し、３２ビットとされたデータが加算器５４ｆに入力される。

【００６３】

さらに、エクステンションが“０１１”である場合、１サイクル毎に、Ｉレジスタ５４ａの下位１６ビットと上位１６ビットとを交互に選択し、３２ビットに符号拡張されたデータが加算器５４ｆに入力される。

続いて、ポスト・レジスタ・アップデート・アドレッシングの場合について説明する。

【００６４】

ポスト・レジスタ・アップデート・アドレッシングの場合、サイクル“１”において、ベースアドレスの値が、ＭＵＸ５４ｃを介してＴレジスタ５４ｂに格納される。

一方、インデックスアドレスの値は、Ｉレジスタ５４ａに格納される。

次いで、サイクル“２”において、Ｔレジスタ５４ｂのベースアドレスの値が、ＭＵＸ５４ｅを介して、そのままターゲットアドレス（ロードアドレス）として出力される。

【００６５】

また、同時に、Ｔレジスタ５４ｂのベースアドレスの値は、加算器５４ｆにも出力され、Ｉレジスタ５４ａに記憶されているインデックスアドレスの値と加算

される。

次いで、加算器５４ｆの加算結果は、ＭＵＸ５４ｃを介して、Ｔレジスタ５４ｂに格納され、Ｔレジスタ５４ｂがベースアドレスとして記憶する値が更新される。

【００６６】

そして、第３サイクル以後は、新たにインデックスアドレスが入力され、サイクル“１”，“２”と同様に、Ｔレジスタ５４ｂの出力をターゲットアドレスとして出力すると共に、Ｔレジスタ５４ｂがベースアドレスとして記憶する値が更新される。

その後、実行されているロード命令の最後の要素データのターゲットアドレスを出力するのと同一のタイミングで、加算器５４ｆの出力がアップデート・ベースアドレスとしてレジスタファイル４０に出力される。

【００６７】

レジスタファイル４０では、ベースレジスタ・ライト信号によって指示されたタイミングで、同時に入力されるベースレジスタ・ライト選択信号によって指定されたレジスタにアップデート・ベースアドレスを記憶する。

このとき、エクステンションフィールドのコードが入力された命令パイプライン制御部５１が、ＭＵＸ５４ｄを制御することにより、図３に示すポスト・レジスタ・アップデート・アドレッシングの拡張仕様が実現される。

【００６８】

即ち、エクステンションが“０００”である場合、Ｉレジスタ５４ａに格納されたデータ（ここでは３２ビット）が、そのまま加算器５４ｆに入力される。

一方、エクステンションが“００１”である場合、Ｉレジスタ５４ａの下位１６ビットを符号拡張し、３２ビットとされたデータが加算器５４ｆに入力される。

【００６９】

また、エクステンションが“０１０”である場合、Ｉレジスタ５４ａの上位１６ビットを符号拡張し、３２ビットとされたデータが加算器５４ｆに入力される。

さらに、エクステンションが“０１１”である場合、１サイクル毎に、Ｉレジスタ５４ａの下位１６ビットと上位１６ビットとを交互に選択し、３２ビットに符号拡張されたデータが加算器５４ｆに入力される。

【００７０】

また、エクステンションが“１００”～“１１１”までである場合、エクステンションが“０００”～“０１１”までの場合とそれぞれ同様の機能拡張を行うが、命令パイプライン制御部５１がベースレジスタ・ライト信号を出力することなく、ベースレジスタの更新を行わないように制御する。

以上のように、本実施の形態に係るベクトルプロセッサ１は、インデックスによって指定されたベクトルレジスタの要素レジスタあるいはスカラーレジスタを複数の領域に分割し、分割した領域のいずれかを選択することによって、所定のインデックスベクトル（指標ベクトル）を取得することとしている。

【００７１】

したがって、１つのベクトルレジスタに、実質的に複数のインデックスベクトルを格納できることとなり、レジスタリソースを効率的に使用することが可能となる。

また、インデックスベクトルを用意する手順としては、１つのインデックスベクトル（指標ベクトル）の場合と同様であるため、プログラムのコードサイズや処理サイクルがほぼ増加することがない。

【００７２】

さらに、ポスト・レジスタ・アップデート・アドレッシングの場合、インデックスレジスタとしてスカラーレジスタが指定され、そのレジスタを複数の領域に分割し、分割した領域のいずれかを選択することによって、所定のインデックスベクトル（指標ベクトル）を取得することとしている。

そのため、インデックスレジスタとしてベクトルレジスタを用いる場合、ベクトルレジスタ全てにデータを格納した後でなければ後段の処理を開始できないのに対し、スカラーレジスタを使用できることから、スカラーレジスタにデータを格納した後、直ちに後段の処理が開始できる。

【００７３】

したがって、インデックスベクトル（指標ベクトル）を用意する手順が軽減され、プログラムのコードサイズや処理サイクル数を減少させることが可能となる。

即ち、本発明によれば、間接指標ベクトル参照をより効率的に行うことが可能となる。

【００７４】

なお、本実施の形態においては、アドレッシングの機能拡張のために、レジスタを上位および下位の２つの領域に分割して使用することとして説明したが、３つ以上の領域に分けて使用することも可能である。

さらに、本実施の形態においては、命令コードに含まれるエクステンションによって、その命令全体における要素レジスタのアドレスを固定的に指定する（単一のレジスタを指定したり、２つのレジスタを交互に指定する等）こととして説明したが、命令実行中に、要素レジスタの指定パターンを動的に変化させることも可能である。即ち、エクステンションのデータ長には限りがあり、複雑なパターンを指定できないことから、複雑な指定パターンを入力可能な手段を備えることにより、エクステンションによって指定可能なパターンより複雑なパターンを指定することが可能である。また、命令実行中に、要素レジスタを指定するための異なる指定パターンを入力したりすることも可能である。

（第２の実施の形態）

本実施の形態に係るベクトルプロセッサは、第１の実施の形態に係るベクトルプロセッサにおいて、図２および図３に示すアドレス修飾の拡張仕様をさらに発展させた仕様となっている。

【００７５】

具体的には、命令コードにアドレッシングの機能拡張のためのコードを含め、インデックスレジスタの分割した領域を多様に指定することに加え、アドレッシングの機能拡張のための指標ベクトルを記憶する２つのレジスタ（以下、「サブインデックスレジスタＡ，Ｂ」と言う。）を指定対象としてさらに設けている。そして、命令コードのエクステンションのうち４ビットを使用して、アドレス修飾の方法を指定することにより、多様なアドレス修飾を可能としている。

【００７６】

したがって、初めに、本実施の形態に係るベクトルプロセッサの拡張仕様について具体的に説明する。

図７は、インデックス修飾アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。なお、図７においては、アドレス修飾の拡張仕様を説明するため、Ｃ言語に準じた記述スタイルのプログラミング例を併せて示しており、後述の図８においても同様である。

【００７７】

図７において、エクステンションが“００００”である場合、インデックスとして指定されたレジスタの内容（指標ベクトル）を符号付きで解釈し、アドレス修飾を行う。

具体的には、ベースに示されたレジスタの値を常に基準とし、その値にインデックスに示されるアドレスを順次加算することにより、アドレス修飾を行う。

【００７８】

また、エクステンションが“０１００”である場合、インデックスとして指定されたレジスタの下位１６ビットを符号付きで解釈し、アドレス修飾を行う。

さらに、エクステンションが“０１１０”である場合、インデックスとして指定されたレジスタの上位１６ビットを符号付きで解釈し、アドレス修飾を行う。

次に、ポスト・レジスタ・アップデート・アドレッシングの拡張仕様について説明する。

【００７９】

図８は、ポスト・レジスタ・アップデート・アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

図８において、エクステンションが“００００”、“０１００”、“０１０１”、“０１１０”および“０１１１”である場合、図３における“０００”、“００１”、“１０１”、“０１０”、“１１０”とそれぞれ同様にアドレスの更新を行う。

【００８０】

また、エクステンションが“１１００”である場合、インデックスとして指定されたインデックスレジスタおよびサブインデックスレジスタＡにおいて、下位１６ビットあるいは上位１６ビットのうち所定のデータを符号付きで解釈し、アドレスの更新を行う。このとき、所定のレジスタ（以下、「インデックススイッチレジスタ」と言う。）に記憶された指定パターンに基づいて、インデックスレジスタおよびサブインデックスレジスタＡの上位１６ビットあるいは下位１６ビットのいずれかが指定される。

【００８１】

具体的には、ベースに指定されたレジスタの値を所定のレジスタ（図８中のパラメータＴの値を格納するレジスタ）に一旦格納し、そのレジスタの値を更新しながらアドレス修飾を行う。

また、エクステンションが“１１１０”の場合、インデックスとして指定されたインデックスレジスタおよびサブインデックスレジスタＢにおいて、下位１６ビットあるいは上位１６ビットのうち所定のデータを符号付きで解釈し、アドレスの更新を行う。このとき、インデックススイッチレジスタに記憶された指定パターン（後述の「レジスタスイッチングパターン」）に基づいて、インデックスレジスタおよびサブインデックスレジスタＢの上位１６ビットあるいは下位１６ビットのいずれかが指定される。

【００８２】

ここで、エクステンションが“１１００”および“１１１０”の場合、ベクトル命令における命令の繰り返し回数の間、インデックスとして指定された内容に応じてアドレス修飾が行われ、繰り返し回数が終了した後、ベースに指定されたレジスタの値が更新される。

したがって、複数のロード命令あるいはストア命令を用いて、繰り返しロードあるいはストアを行う場合、ベースに指定されたレジスタが自動的に更新されるので、命令発行の都度、ベースのアドレスを別の命令で更新する必要がなく、直ちに命令の実行を行うことが可能である。

【００８３】

また、図８において、エクステンションが“１１０１”および“１１１１”の場合、エクステンションが“１１００”および“１１１０”の場合とそれぞれ同様にアドレスの更新を行う。ただし、エクステンションが“１１０１”および“１１１１”の場合、命令の繰り返し回数が終了した後、ベースに指定されたレジスタの値を更新することなく命令の実行を終了する。

【００８４】

次に、本実施の形態に係るベクトルプロセッサの構成を説明する。

図９は、本実施の形態に係るベクトルプロセッサ２の構成を示す図である。

図９において、ベクトルプロセッサ２の構成は、レジスタファイル１４０にサブインデックスレジスタ１４１，１４２とインデックススイッチレジスタ１４３とが含まれることを除き、第１の実施の形態におけるベクトルプロセッサ１の構成と同様である。したがって、サブインデックスレジスタ１４１，１４２およびインデックススイッチレジスタ１４３についてのみ説明し、他の部分については、図４において対応する部分を参照することとする。

【００８５】

サブインデックスレジスタ１４１，１４２は、レジスタファイル１４０に含まれるスカラーレジスタあるいはベクトルレジスタ（ここでは、スカラーレジスタＳＲ３０，３１とする）によって構成される。そして、サブインデックスレジスタ１４１，１４２には、所定の指標ベクトルがそれぞれ格納されている。

インデックススイッチレジスタ１４３は、レジスタファイル１４０に含まれるスカラーレジスタ（ここでは、スカラーレジスタＳＲ２９とする）によって構成され、インデックスレジスタおよびサブインデックスレジスタ１４１，１４２に含まれるいずれのデータをインデックスアドレスとして選択するかを示すデータ（以下、「レジスタスイッチングパターン」と言う。）を記憶している。

【００８６】

レジスタスイッチングパターンは、具体的には、サブインデックスレジスタ１４１，１４２のいずれか一方における上位１６ビットおよび下位１６ビットと、後述する図１１のＩレジスタにおける上位１６ビットおよび下位１６ビットとを含む４つの領域のうち、アドレス修飾を行う要素ごとに、１つの領域を選択する

ためのデータである。

【００８７】

図１０は、レジスタスイッチングパターンのデータ構成を示す図である。

図１０において、レジスタスイッチングパターンの上位半分には、サブインデックスレジスタ１４１が選択された場合のスイッチングパターンが格納され、下位半分には、サブインデックスレジスタ１４２が選択された場合のスイッチングパターンが格納されている。

【００８８】

次に、図１１は、ロードユニット１５０の内部構成を示すブロック図である。

図１１において、ロードユニット１５０の内部構成は、命令パイプライン制御部１５１およびアドレス演算回路１５４の機能を除き、第１の実施の形態におけるロードユニット５０の内部構成と同様である。したがって、命令パイプライン制御部１５１およびアドレス演算回路１５４についてのみ説明し、他の部分については、図５において対応する部分の説明を参照することとする。

【００８９】

命令パイプライン制御部１５１は、ロードユニット１５０全体を制御するものである。また、命令パイプライン制御部１５１は、命令コードに含まれるエクステンションフィールドのコードに応じて、サブインデックスレジスタ１４１，１４２のいずれかのサブインデックスアドレスをアドレス演算回路１５４（より詳細にはＭＵＸ１５４ｇ）に選択させると共に、レジスタファイル１４０のインデックススイッチレジスタ１４３における上位あるいは下位半分のレジスタスイッチングパターンを後述するＪレジスタ１５４ｈに記憶させる。

【００９０】

アドレス演算回路１５４は、命令パイプライン制御部１５１の指示に基づいて、レジスタファイル１４０から入力される、ベースアドレスおよびインデックスアドレス（インデックスフィールドによって指定されたアドレス）と、サブインデックスレジスタ１４１，１４２に記憶されたアドレス（以下、「サブインデックスアドレス」と言う。）と、インデックススイッチレジスタ１４３に記憶されたレジスタスイッチングパターンとから、ロード命令の対象となるメモリ１１０

上のアドレス（ロードアドレス）を算出する。

【００９１】

続いて、図１１におけるアドレス演算回路１５４の構成について説明する。

図１２は、アドレス演算回路１５４の構成例を示す図である。

図１２において、アドレス演算回路１５４の構成は、マルチプレクサ（ＭＵＸ）１５４ｄ，１５４ｇと、Ｊレジスタ１５４ｈと、シフタ１５４ｉとを除き、第１の実施の形態におけるアドレス演算回路５４の構成と同様である。したがって、ＭＵＸ１５４ｄ，１５４ｇと、Ｊレジスタ１５４ｈと、シフタ１５４ｉとについてのみ説明し、他の部分については、図６を参照において対応する部分を参照することとする。

【００９２】

ＭＵＸ１５４ｄは、Ｉレジスタ１５４ａから入力されるインデックスアドレスの上位あるいは下位のアドレスと、後述するＭＵＸ１５４ｇから入力されるサブインデックスアドレスの上位あるいは下位のアドレスのいずれかを選択して加算器１５４ｆに出力する。このとき、ＭＵＸ１５４ｄは、後述するシフタ１５４ｉから入力される指示信号に基づいて、入力されたインデックスアドレスあるいはサブインデックスアドレスの上位あるいは下位のアドレスのいずれかを選択する。

【００９３】

ＭＵＸ１５４ｇは、サブインデックスレジスタ１４１，１４２から入力されるサブインデックスアドレスのうち、いずれかを選択してＭＵＸ５４ｄに出力する。このとき、ＭＵＸ１５４ｇは、命令パイプライン制御部１５１から入力される選択指示信号に基づいて、いずれかのサブインデックスアドレスを選択する。

Ｊレジスタ１５４ｈは、レジスタファイル１４０から入力されるインデックレジスタ選択データを一時的に記憶する。

【００９４】

シフタ１５４ｉは、Ｊレジスタ１５４ｈからレジスタスイッチングパターンを受け取り、１サイクル毎に、レジスタスイッチングパターンに基づいて、ＭＵＸ５４ｄが選択するアドレスを指示するための信号を出力する。

次に、動作を説明する。

初めに、図９を参照して、ベクトルプロセッサ２全体の動作について説明する。

【００９５】

ベクトルプロセッサ２において処理が行われる場合、メモリ制御部１２０を介してメモリ１１０から命令フェッチ部１３０に命令コードが読み出される。

そして、ロードユニット１５０、ストアユニット１６０および演算ユニット１７０のそれぞれに、命令フェッチ部１３０から命令コードが出力される。

命令コードが入力されたロードユニット１５０、ストアユニット１６０および演算ユニット１７０は、その命令コードをデコードし、それぞれのユニットに対応する命令である場合にのみ、命令を実行する。

【００９６】

ここでは、図１１を参照しつつ、命令コードがロード命令である場合について説明する。

命令コードのオペコードがロード命令を示している場合（より詳細にはprefixコードが“０００”である場合）、ロードユニット１５０が動作する。

まず、ロードユニット１５０は、命令フェッチ部１３０から受け取ったベースフィールドのコードを、ベースレジスタ・リード選択信号（データの読み出し時にベースレジスタを選択する信号）としてレジスタファイル１４０に出力する。

【００９７】

そして、ベースレジスタ・リード選択信号を受け取ったレジスタファイル１４０から、ベースレジスタ・リード選択信号によって指定されたレジスタに記憶されたベースアドレスの値がロードユニット１５０に入力される。

レジスタファイル１４０から入力されたベースアドレスの値は、アドレス演算回路１５４に入力される。

【００９８】

また、インデックスレジスタ決定回路１５２は、命令フェッチ部１３０から入力されたインデックスフィールドのコードおよびリピートアマウントフィールドのコードを受け取る。そして、インデックスレジスタ決定回路１５２は、命令パ

イプライン制御部１５１からの指示に従って、命令コードがベクトル命令であるかスカラー命令であるかを判定し、ベクトル命令である場合、インデックスレジスタ選択信号を、リピートアマウントに示される要素データ数分、レジスタファイル１４０に順次出力する。このとき、インデックスレジスタ選択信号によって指定されたレジスタがベクトルレジスタである場合、指定されたベクトルレジスタ内の各要素レジスタを特定するための所定の選択信号が出力される。

【００９９】

インデックスレジスタ選択信号が入力されたレジスタファイル１４０からは、インデックスレジスタ選択信号に示されるレジスタから、インデックスアドレスの値が、アドレス演算回路１５４に順次入力される。

また、命令パイプライン制御部１５１は、レジスタファイル１４０に、サブインデックスアドレス１４１，１４２およびインデックススイッチレジスタ１４３のアドレスを出力する。そして、インデックスアドレスの値がアドレス演算回路１５４に入力されるタイミングに合わせて、サブインデックスアドレスの値も、アドレス演算回路１５４に順次入力される。

【０１００】

インデックスアドレスおよびサブインデックスアドレスの値を受け取ったアドレス演算回路１５４は、レジスタスイッチングパターンに基づいて、インデックスアドレスおよびサブインデックスアドレスのうち、所定の領域を選択し、ベースアドレスの値と、選択した領域に記憶されているアドレスの値とからロードアドレスを算出し、メモリ制御部１２０に出力する。なお、アドレス演算回路１５４の動作については後述する。

【０１０１】

また、命令フェッチ部１３０から入力されたデスティネーションフィールドのコードおよびリピートアマウントフィールドのコードは、パイプライン処理におけるタイミングを合わせるべく、ＰＲ１５５，１５６に一旦記憶された後、デスティネーションレジスタ決定回路１５３に入力される。

すると、デスティネーションレジスタ決定回路１５３は、デスティネーションレジスタ選択信号を、メモリ１１０からロードされたデータと同期させてレジス

タファイル１４０に出力する。

【０１０２】

そして、レジスタファイル１４０において、メモリ１１０からロードされたデータが、所定のデスティネーションレジスタに順次記憶される。

さらに、ポスト・レジスタ・アップデート・アドレッシングの場合、アドレス演算回路１５４によって出力されるアップデート・ベースアドレスをベースレジスタに書き込む。

【０１０３】

したがって、ベースフィールドのコードをレジスタ１５７に保持しておき、レジスタ１５７によって出力されるコードをベースレジスタ・ライト選択信号（データの書き込み時にベースレジスタを選択する信号）として用い、ベースレジスタ・ライト信号（ベースレジスタに対する書き込み指示信号）が入力されることに対応して、ベースレジスタのデータが更新される。

【０１０４】

次に、アドレス演算回路１５４の動作について説明する。

まず、インデックス修飾アドレッシングの場合について説明する。

インデックス修飾アドレッシングの場合、サイクル“1”において、ベースアドレスの値が、ＭＵＸ１５４ｃを介してＴレジスタ１５４ｂに格納される。

一方、インデックスアドレスの値は、Ｉレジスタ１５４ａに格納され、レジスタスイッチングパターンの上位あるいは下位半分のデータが、Ｊレジスタ１５４ｈに格納される。

【０１０５】

次いで、サイクル“2”において、Ｊレジスタ１５４ｈに格納されているレジスタスイッチングパターン（上位あるいは下位半分）における最下位２ビットがシフタ１５４ｉによってＭＵＸ１５４ｄに出力される。そして、ＭＵＸ１５４ｄは、シフタ１５４ｉから入力された２ビットのデータに基づいて、ＭＵＸ１５４ｇを介して入力されたサブインデックスアドレスの上位あるいは下位いずれかの１６ビットのデータを選択し、加算器１５４ｆに出力する。また、Ｔレジスタ１５４ｂのベースアドレスの値と、ＭＵＸ１５４ｄから出力されたアドレスの値と

が、加算器１５４ｆによって加算され、ＭＵＸ１５４ｅを介してターゲットアドレス（ロードアドレス）として出力される。

【０１０６】

そして、第３サイクル以後は、ベースアドレスの値がＴレジスタ１５４ｂに保持された状態のまま、シフタ１５４ｉによって、Ｊレジスタ１５４ｈに格納されたレジスタスイッチングパターン（上位あるいは下位半分）において、下位から２ビットずつがＭＵＸ１５４ｄに順次出力され、加算器１５４ｆによって、ＭＵＸ１５４ｄから出力されるアドレスが、ベースアドレスと順次加算される。

【０１０７】

このとき、エクステンションフィールドのコードが入力された命令パイプライン制御部１５１が、ＭＵＸ１５４ｇを制御すると共に、エクステンションフィールドのコードに応じて、レジスタスイッチングパターンの上位あるいは下位半分のデータのいずれかを選択してＪレジスタ１５４ｈに記憶させることにより、図７に示すインデックス修飾アドレッシングの拡張仕様が実現される。

【０１０８】

即ち、エクステンションが“００００”である場合、Ｉレジスタ１５４ａに格納されたデータ（ここでは３２ビット）がＭＵＸ１５４ｄによって選択され、そのまま加算器１５４ｆに入力される。

一方、エクステンションが“０１００”である場合、Ｉレジスタ１５４ａの下位１６ビットを符号拡張し、３２ビットとされたデータがＭＵＸ１５４ｄによって選択され、加算器１５４ｆに入力される。

【０１０９】

また、エクステンションが“０１１０”である場合、Ｉレジスタ１５４ｄの上位１６ビットを符号拡張し、３２ビットとされたデータがＭＵＸ１５４ｄによって選択され、加算器１５４ｆに入力される。

続いて、ポスト・レジスタ・アップデート・アドレッシングの場合について説明する。

【０１１０】

ポスト・レジスタ・アップデート・アドレッシングの場合、サイクル“１”に

おいて、ベースアドレスの値が、ＭＵＸ１５４ｃを介してＴレジスタ１５４ｂに格納される。

一方、インデックスアドレスの値は、Ｉレジスタ１５４ａに格納され、レジスタスイッチングパターンの上位あるいは下位半分のデータが、命令パイプライン制御部１５１の指示に従ってＪレジスタ１５４ｈに格納される。

【０１１１】

次いで、サイクル“2”において、Ｔレジスタ１５４ｂのベースアドレスの値が、ＭＵＸ１５４ｅを介して、そのままターゲットアドレス（ロードアドレス）として出力される。

また、同時に、Ｔレジスタ１５４ｂのベースアドレスの値は、加算器１５４ｆにも出力され、ＭＵＸ１５４ｄから出力されたアドレスの値と加算される。

【０１１２】

次いで、加算器１５４ｆの加算結果は、ＭＵＸ１５４ｃを介して、Ｔレジスタ１５４ｂに格納され、Ｔレジスタ１５４ｂがベースアドレスとして記憶する値が更新される。

そして、第３サイクル以後は、シフタ１５４ｉによって、Ｊレジスタ１５４ｈに格納されたレジスタスイッチングパターン（上位あるいは下位半分）において、下位から２ビットずつがＭＵＸ１５４ｄに順次出力され、サイクル“1”，“2”と同様に、Ｔレジスタ１５４ｂの出力をターゲットアドレスとして出力すると共に、Ｔレジスタ１５４ｂがベースアドレスとして記憶する値が更新される。

【０１１３】

その後、実行されているロード命令の最後の要素データのターゲットアドレスを出力するのと同一のタイミングで、加算器１５４ｆの出力がアップデート・ベースアドレスとしてレジスタファイル１４０に出力される。

レジスタファイル１４０では、ベースレジスタ・ライト信号によって指示されたタイミングで、同時に入力されるベースレジスタ・ライト選択信号によって指定されたレジスタにアップデート・ベースアドレスを記憶する。

【０１１４】

このとき、エクステンションフィールドのコードが入力された命令パイプライ

ン制御部１５１が、ＭＵＸ１５４ｇを制御すると共に、エクステンションフィールドのコードに応じて、レジスタスイッチングパターンの上位あるいは下位半分のデータのいずれかを選択してＪレジスタ１５４ｈに記憶させることにより、図８に示すポスト・レジスタ・アップデート・アドレッシングの拡張仕様が実現される。

【０１１５】

即ち、エクステンションが“０００００”である場合、Ｉレジスタ１５４ａに格納されたデータ（ここでは３２ビット）がＭＵＸ１５４ｄによって選択され、そのまま加算器１５４ｆに入力される。

一方、エクステンションが“０１００”である場合、Ｉレジスタ１５４ａに格納されたデータの下位１６ビットがＭＵＸ１５４ｄによって選択され、符号拡張して３２ビットとされたデータが加算器１５４ｆに入力される。

【０１１６】

また、エクステンションが“０１１０”である場合、Ｉレジスタ１５４ａに格納されたデータの上位１６ビットがＭＵＸ１５４ｄによって選択され、符号拡張して３２ビットとされたデータが加算器１５４ｆに入力される。

また、エクステンションが“１１００”である場合、Ｉレジスタ１５４ａに格納されたデータおよびサブインデックスレジスタ１４１に格納されたデータそれぞれにおいて、下位１６ビットあるいは上位１６ビットのうち、レジスタスイッチングパターンに示されるデータがＭＵＸ１５４ｄによって順次選択され、符号拡張して３２ビットとされたデータが加算器１５４ｆに入力される。

【０１１７】

また、エクステンションが“１１１０”である場合、Ｉレジスタ１５４ａに格納されたデータおよびサブインデックスレジスタ１４２に格納されたデータそれぞれにおいて、下位１６ビットあるいは上位１６ビットのうち、レジスタスイッチングパターンに示されるデータがＭＵＸ１５４ｄによって順次選択され、符号拡張して３２ビットとされたデータが加算器１５４ｆに入力される。

【０１１８】

また、エクステンションが“０１０１”、“０１１１”、“１１０１”および

“１１１１”である場合、エクステンションが“０１００”、“０１１０”、“１１００”および“１１１０”の場合とそれぞれ同様の機能拡張を行うが、命令パイプライン制御部１５１がベースレジスタ・ライト信号を出力することなく、ベースレジスタの更新を行わないように制御する。

【０１１９】

以上のように、本実施の形態に係るベクトルプロセッサ２は、インデックスによって指定されたベクトルレジスタの要素レジスタあるいはスカラーレジスタを複数の領域に分割すると共に、指標ベクトルを格納した他のレジスタ（サブインデックスレジスタ１４１，１４２）を用意し、このレジスタも同様に、複数の領域に分割して使用する。そして、インデックスによって指定されたレジスタおよび用意した他のレジスタにおいて、分割した領域のいずれかを選択することによって、所定のインデックスベクトル（指標ベクトル）を取得することとしている。

【０１２０】

したがって、少数のレジスタに、実質的に、より多くのインデックスベクトルを格納できることとなり、レジスタリソースを効率的に使用することが可能となる。

また、ベクトルプロセッサ２は、インデックススイッチレジスタ１４３を備えている。そして、インデックススイッチレジスタ１４３に記憶されたレジスタスイッチングパターンによって、アドレス修飾を行う要素毎に、インデックスレジスタおよびサブインデックスレジスタ１４１，１４２の上位あるいは下位の領域のいずれかを任意に指定することが可能である。

【０１２１】

したがって、１つの命令を実行する際に、インデックスレジスタおよびサブインデックスレジスタ１４１，１４２における領域のいずれを選択するかを動的に変化させることができ、領域を指定することによって得られるインデックスベクトルを、より多様なものとすることができる。

即ち、本発明によれば、間接指標ベクトル参照をより効率的に行うことが可能となる。

【０１２２】

なお、本実施の形態においては、レジスタスイッチングパターンをインデックススイッチレジスタ１４３に格納しておくこととして説明したが、レジスタスイッチングパターンを命令コードのインデックスフィールドに含めることとしてもよい。

図１３は、レジスタスイッチングパターンを命令コードのインデックスフィールドに含める場合のデータ構成を示す図である。

【０１２３】

図１３において、インデックスフィールドには、レジスタスイッチングパターンと、第１および第２のインデックスが格納されたそれぞれの領域（例えば、インデックスレジスタの上位および下位１６ビット）を示すアドレスとが含まれている。

そして、アドレス修飾を行う８個の要素それぞれについて、２つのインデックスのいずれによって、アドレス修飾を行うかが任意に指定されている。

【０１２４】

また、この場合、図１２のアドレス演算回路１５４の構成は、以下のようになる。

図１４は、レジスタスイッチングパターンを命令コードのインデックスフィールドに含める場合のアドレス演算回路１５４の構成を示す図である。

図１４においては、レジスタスイッチングパターンが命令コードに含まれていることから、インデックススイッチレジスタ１４３に相当するレジスタが設けられておらず、レジスタファイル１４０を介して、Ｉレジスタ１５４ａにレジスタスイッチングパターンと、第１および第２のインデックスが格納されたそれぞれの領域のアドレスとが入力される。

【０１２５】

そして、入力されたレジスタスイッチングパターンは、シフタ１５４ｉに入力され、アドレス修飾を行う要素毎に、Ｉレジスタ１５４ａに記憶された２つのインデックスのいずれかを選択するための信号をＭＵＸ１５４ｄに出力する。

このような構成とすることにより、インデックススイッチレジスタ１４３を設

けることなく、領域を指定することによって得られるインデックスベクトルを、多様なものとすることが可能となる。

【図面の簡単な説明】

【図１】　ロード命令あるいはストア命令のフォーマットを示す図である。

【図２】　インデックス修飾アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

【図３】　ポスト・レジスタ・アップデート・アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

【図４】　本実施の形態に係るベクトルプロセッサ１の構成を示す図である。

【図５】　ロードユニット５０の内部構成を示すブロック図である。

【図６】　アドレス演算回路５４の構成例を示す図である。

【図７】　インデックス修飾アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

【図８】　ポスト・レジスタ・アップデート・アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

【図９】　本実施の形態に係るベクトルプロセッサ２の構成を示す図である。

【図１０】　レジスタスイッチングパターンのデータ構成を示す図である。

【図１１】　ロードユニット１５０の内部構成を示すブロック図である。

【図１２】　アドレス演算回路１５４の構成例を示す図である。

【図１３】　レジスタスイッチングパターンを命令コードのインデックスフィールドに含める場合のデータ構成を示す図である。

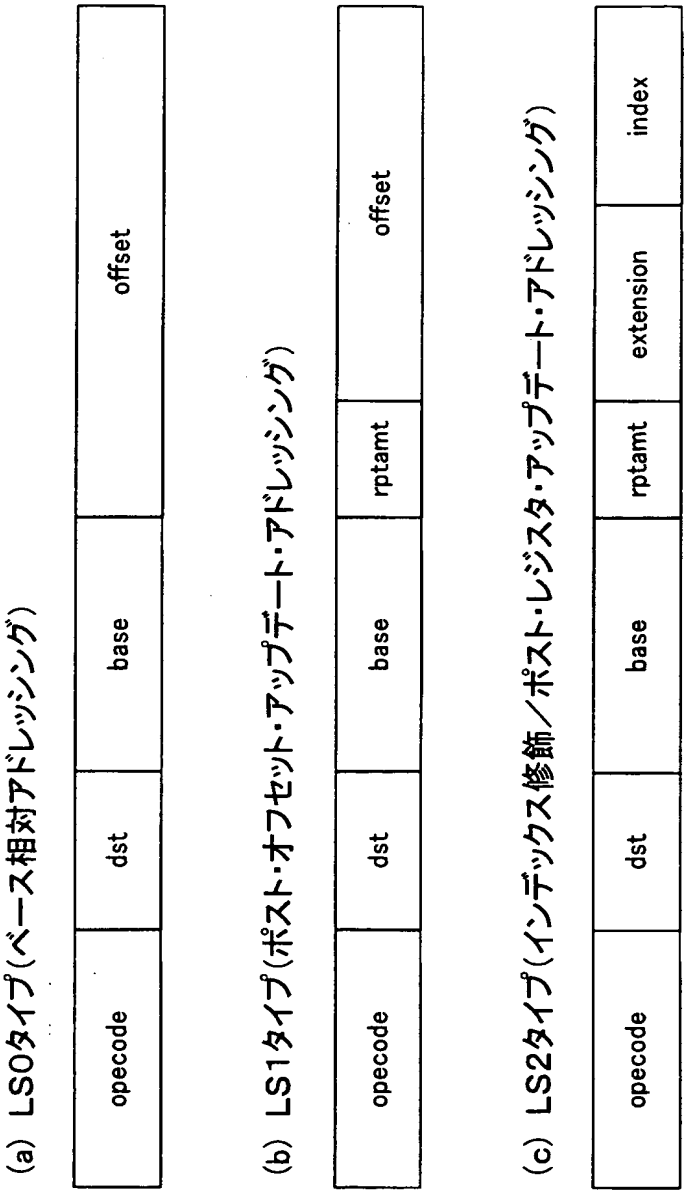
【図１４】　レジスタスイッチングパターンを命令コードのインデックスフィールドに含める場合のアドレス演算回路１５４の構成を示す図である。

【符号の説明】

１，２　ベクトルプロセッサ、１０，１１０　メモリ，２０，１２０　メモリ制御部，３０，１３０　命令フェッチ部，４０，１４０　レジスタファイル，５０，１５０　ロードユニット，５１，１５１　命令パイプライン制御部，５２，１５２　インデックスレジスタ決定回路，５３，１５３　デスティネーションレジスタ決定回路，５４，１５４　アドレス演算回路，５４ａ，１５４ａ　Ｉレジスタ，５４ｂ，１５４ｂ　Ｔレジスタ，５４ｃ～５４ｅ，１５４ｃ～１５４ｅ，１５４ｇ　ＭＵＸ（マルチプレクサ），５４ｆ，１５４ｆ　加算器，１４５ｈ　Ｊレジスタ，１５４ｉ　シフタ，５５，５６，１５５，１５６　ＰＲ（パイプラインレジスタ），５７，１５７　レジスタ，６０，１６０　ストアユニット，７０，１７０　演算ユニット

【書類名】　図面

【図１】


(a) LS0タイプ(ベース相対アドレッシング)
opecode
dst
base
offset
(b) LS1タイプ(ポスト・オフセット・アップデート・アドレッシング)
opecode
dst
base
rptamt
offset
(c) LS2タイプ(インデックス修飾／ポスト・レジスタ・アップデート・アドレッシング)
opecode
dst
base
rptamt
extension
index

【図２】

extension	アドレス修飾の方法

000	インデックスとして指定されたレジスタの内容を符号付きで解釈し、アドレス修飾を行う。
	for(t=0; t < rptamt; t++){
	　targetAdrs = base.uw + index[t].h[LOWER];
	}

001	インデックスとして指定されたレジスタの下位16ビット(index[t].h[LOWER])を符号付きで解釈し、
	アドレス修飾を行う。
	for(t=0; t < rptamt; t++){
	　targetAdrs = base.uw + index[t].h[LOWER];
	}

010	インデックスとして指定されたレジスタの上位16ビット(index[t].h[UPPER])を符号付きで解釈し、
	アドレス修飾を行う。
	for(t=0; t < rptamt; t++){
	　targetAdrs = base.uw + index[t].h[UPPER];
	}

011	インデックスとして指定されたレジスタの下位16ビット(index[t].h[LOWER])および上位16ビット(index[t].h[UPPER])を
	それぞれ符号付きで解釈し、交互にアドレス修飾を行う。
	for(t=0; t < rptamt; t++){
	　if ((t & 1)==0)　targetAdrs = base.uw + index[t].h[LOWER];
	　else　　targetAdrs = base.uw + index[t].h[UPPER];
	}

【図３】

extension	アドレス修飾の方法

000	インデックスとして指定されたレジスタの内容を符号付きで解釈し、
	アドレスの更新を行う。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].w;
	}
	base.uw = T;

001	インデックスとして指定されたレジスタの下位16ビットの
	index[t].h[LOWER]を符号付きで解釈し、アドレスの更新を行う。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].h[LOWER];
	}
	base.uw = T;

010	インデックスとして指定されたレジスタの上位16ビットの
	index[t].h[UPPER]を符号付きで解釈し、アドレスの更新を行う。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].h[UPPER];
	}
	base.uw = T;

011	インデックスとして指定されたレジスタの下位16ビットの
	index[t].h[LOWER]と上位16ビットのindex[t].h[UPPER]を
	それぞれ符号付きで解釈し、交互にアドレスの更新を行う。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T;
	　if ((t & 1) == 0)　T += index[t].h[LOWER];
	　else　　T += index[t].h[UPPER];
	}
	base.uw = T;

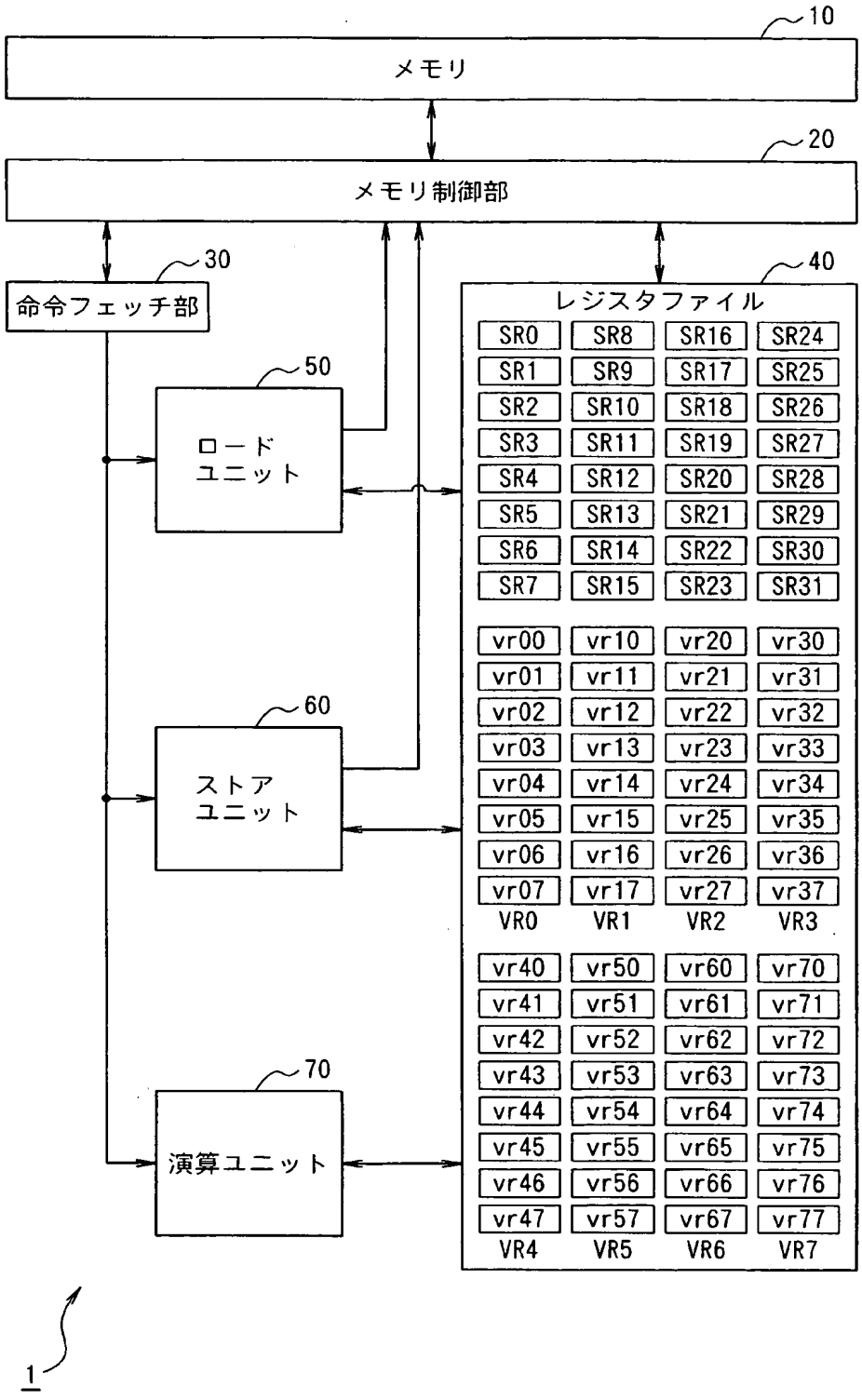
100	インデックスとして指定されたレジスタの内容を符号付きで解釈し、
	アドレスの更新を行う。ベース・レジスタの更新は行わない。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].w;
	}

101	インデックスとして指定されたレジスタの下位16ビットの
	index[t].h[LOWER]を符号付きで解釈し、アドレスの更新を行う。
	ベース・レジスタの更新は行わない。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].h[LOWER];
	}

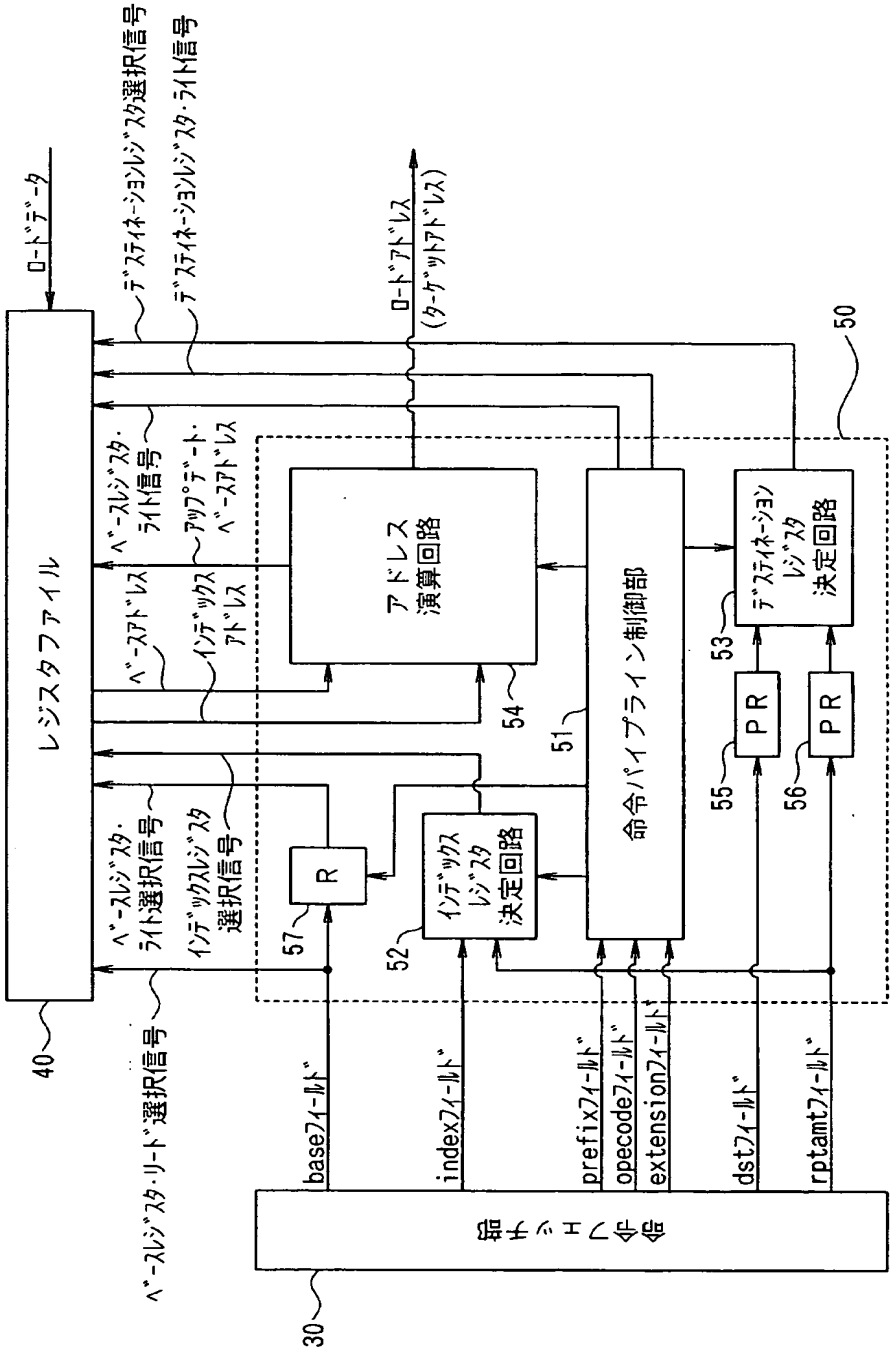
110	インデックスとして指定されたレジスタの上位16ビットの
	index[t].h[UPPER]を符号付きで解釈し、アドレスの更新を行う。
	ベース・レジスタの更新は行わない。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].h[UPPER];
	}

111	インデックスとして指定されたレジスタの下位16ビットの
	index[t].h[LOWER]と上位16ビットのindex[t].h[UPPER]を
	それぞれ符号付きで解釈し、交互にアドレスの更新を行う。
	ベース・レジスタの更新は行わない。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T;
	　if ((t & 1) == 0)　T += index[t].h[LOWER];
	　else　　T += index[t].h[UPPER];
	}

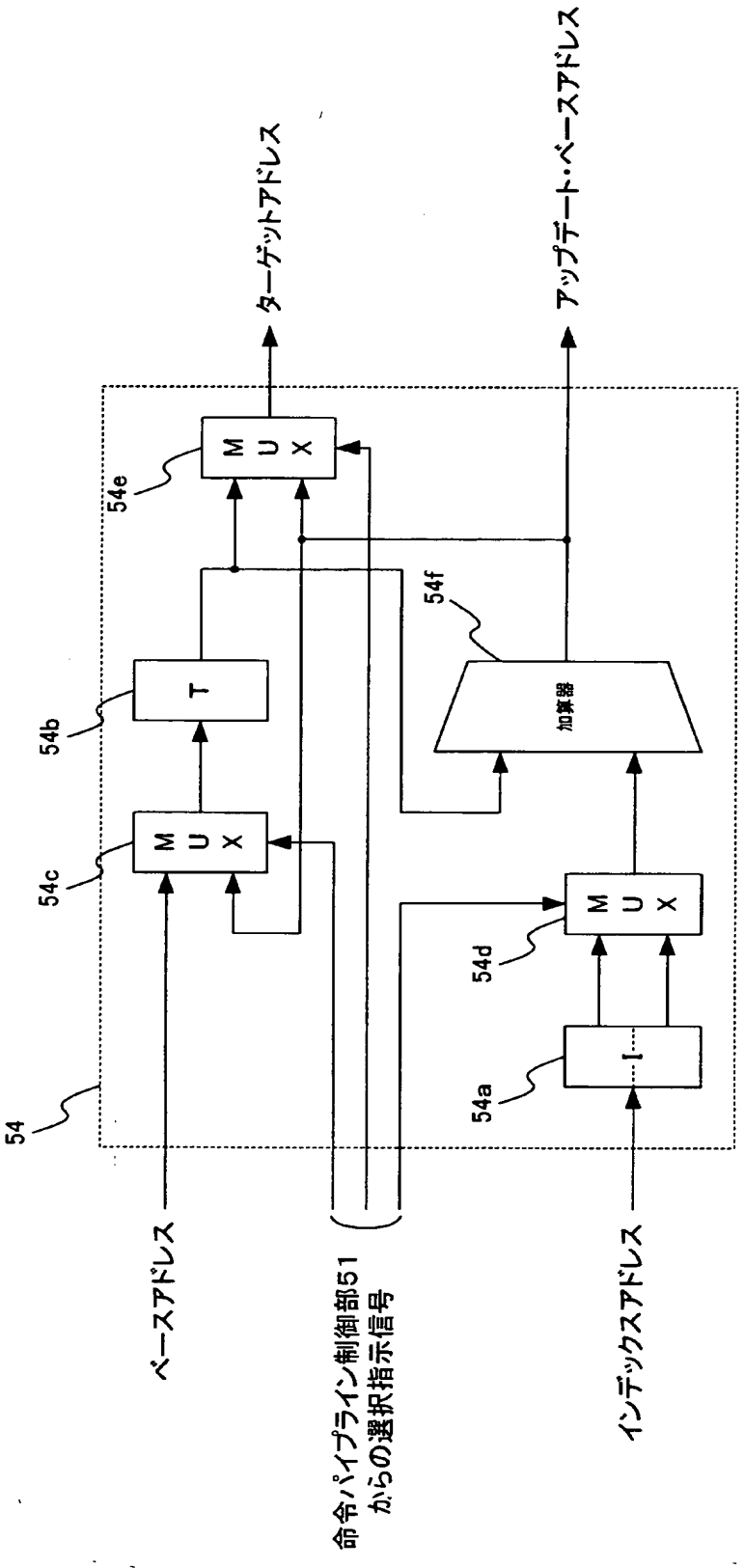
【図４】


10
メモリ
20
メモリ制御部
30
命令フェッチ部
40
レジスタファイル
SR0 SR8 SR16 SR24
SR1 SR9 SR17 SR25
SR2 SR10 SR18 SR26
SR3 SR11 SR19 SR27
SR4 SR12 SR20 SR28
SR5 SR13 SR21 SR29
SR6 SR14 SR22 SR30
SR7 SR15 SR23 SR31
vr00 vr10 vr20 vr30
vr01 vr11 vr21 vr31
vr02 vr12 vr22 vr32
vr03 vr13 vr23 vr33
vr04 vr14 vr24 vr34
vr05 vr15 vr25 vr35
vr06 vr16 vr26 vr36
vr07 vr17 vr27 vr37
VR0 VR1 VR2 VR3
vr40 vr50 vr60 vr70
vr41 vr51 vr61 vr71
vr42 vr52 vr62 vr72
vr43 vr53 vr63 vr73
vr44 vr54 vr64 vr74
vr45 vr55 vr65 vr75
vr46 vr56 vr66 vr76
vr47 vr57 vr67 vr77
VR4 VR5 VR6 VR7
50
ロード
ユニット
60
ストア
ユニット
70
演算ユニット
1

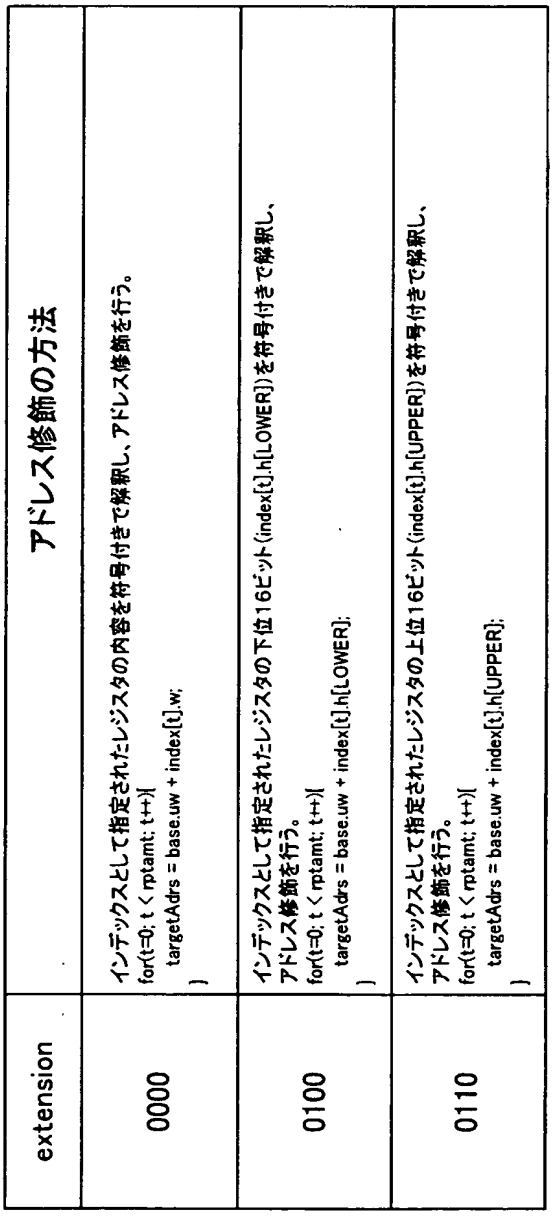
【図５】


ロードデータ
デスティネーションレジスタ選択信号
デスティネーションレジスタ・ライト信号
ロードアドレス
(ターゲットアドレス)
50
レジスタファイル
ベースレジスタ・ライト信号
アップデート・ベースアドレス
ベースアドレス
インデックスアドレス
アドレス演算回路
ベースレジスタ・ライト選択信号
インデックスレジスタ選択信号
R
57
インデックスレジスタ決定回路
52
54
命令パイプライン制御部
51
PR
55
PR
56
デスティネーションレジスタ決定回路
53
40
ベースレジスタ・リード選択信号
baseフィールド
indexフィールド
prefixフィールド
opecodeフィールド
extensionフィールド
dstフィールド
rptamtフィールド
デコーダ部
30

【図６】


ターゲットアドレス
アップデート・ベースアドレス
54e
MUX
54f
加算器
54b
T
54c
MUX
54d
MUX
54a
I
54
ベースアドレス
命令パイプライン制御部51
からの選択指示信号
インデックスアドレス

【図７】



extension	アドレス修飾の方法

0000	インデックスとして指定されたレジスタの内容を符号付きで解釈し、アドレス修飾を行う。
	for(t=0; t < rptamt; t++){
	targetAdrs = base.uw + index[t].w;
	}

0100	インデックスとして指定されたレジスタの下位16ビット(index[t].h[LOWER])を符号付きで解釈し、アドレス修飾を行う。
	for(t=0; t < rptamt; t++){
	targetAdrs = base.uw + index[t].h[LOWER];
	}

0110	インデックスとして指定されたレジスタの上位16ビット(index[t].h[UPPER])を符号付きで解釈し、アドレス修飾を行う。
	for(t=0; t < rptamt; t++){
	targetAdrs = base.uw + index[t].h[UPPER];
	}

【図８】

extension	アドレス修飾の方法

0000	インデックスとして指定したレジスタの内容を符号付きで解釈し、
	アドレスの更新を行う。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].w;
	}
	base.uw = T;

0100	インデックスとして指定したレジスタの下位16ビットの
	index[t].h[LOWER]を符号付きで解釈し、アドレスの更新を行う。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].h[LOWER];
	}
	base.uw = T;

0101	同上、但し、ベース・レジスタの更新を行わない。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].h[LOWER];
	}

0110	インデックスとして指定したレジスタの上位16ビットの
	index[t].h[UPPER]を符号付きで解釈し、アドレスの更新を行う。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].h[UPPER];
	}
	base.uw = T;

0111	同上、但し、ベース・レジスタの更新を行わない。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T; T += index[t].h[UPPER];
	}

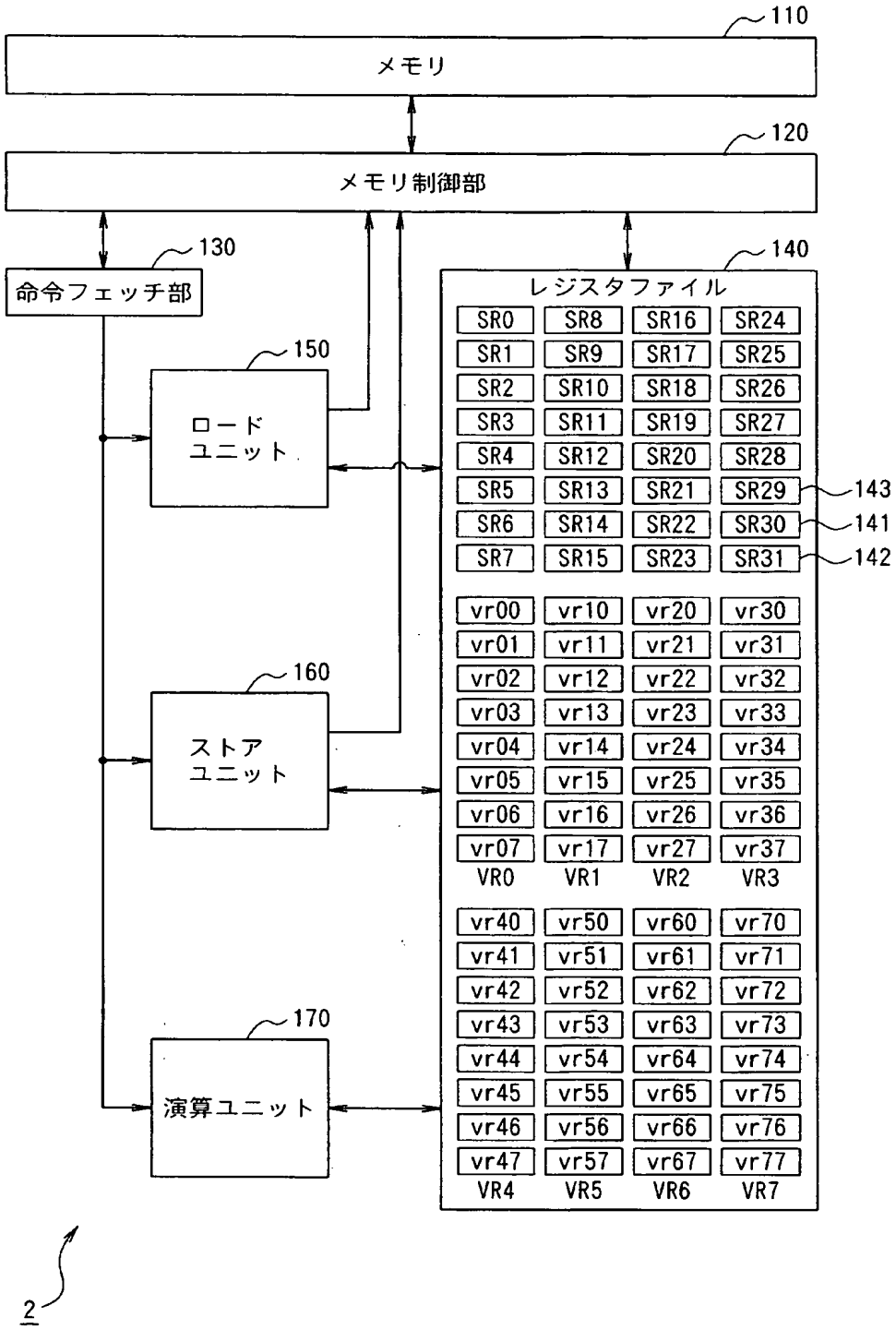
1100	インデックスとして指定したレジスタの下位16ビットの
	index.h[LOWER]と上位16ビットのindex.h[UPPER]および
	サブインデックスレジスタAの下位16ビットのSR30.h[LOWER]および
	上位16ビットのSR30.h[UPPER]のうち所定データを符号付きで解釈し、
	アドレスの更新を行う。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T;
	　switch((ISW.h[LOWER] >> (2 * t)) & 0x3){
	　case 0:　T += index.h[LOWER]; break;
	　case 1:　T += index.h[UPPER]; break;
	　case 2:　T +=SR30.h[LOWER]; break;
	　case 3:　T +=SR30.h[UPPER]; break;
	　}
	}
	base.uw = T;

1101	同上、但し、ベース・レジスタの更新を行わない。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T;
	　switch((ISW.h[UPPER] >> (2 * t)) & 0x3){
	　case 0:　T += index.h[LOWER]; break;
	　case 1:　T += index.h[UPPER]; break;
	　case 2:　T +=SR30.h[LOWER]; break;
	　case 3:　T +=SR30.h[UPPER]; break;
	　}
	}

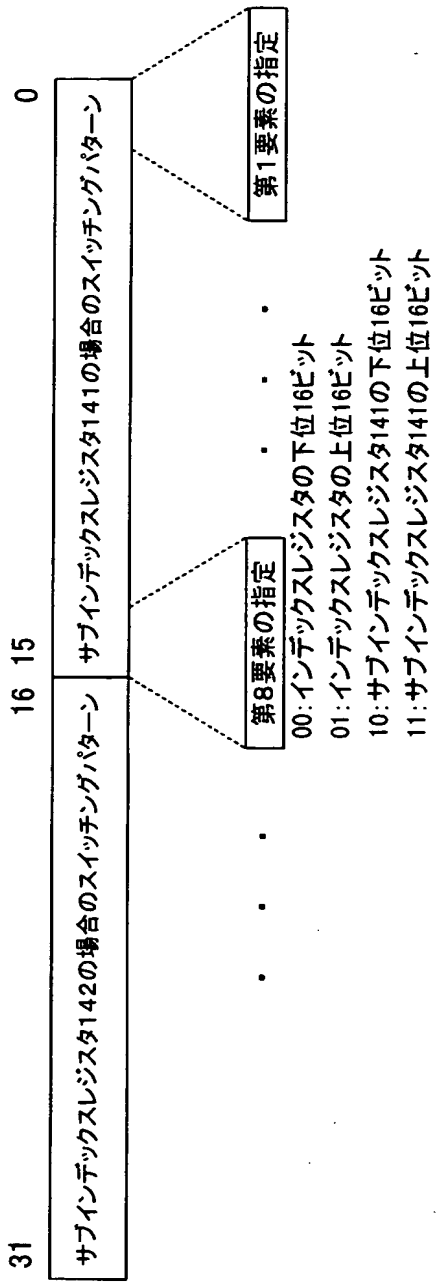
1110	インデックスとして指定したレジスタの下位16ビットの
	index.h[LOWER]と上位16ビットのindex.h[UPPER]および
	サブインデックスレジスタBの下位16ビットのSR31.h[LOWER]および
	上位16ビットのSR31.h[UPPER]のうち所定データを符号付きで解釈し、
	アドレスの更新を行う。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T;
	　switch((ISW.h[LOWER] >> (2 * t)) & 0x3){
	　case 0:　T += index.h[LOWER]; break;
	　case 1:　T += index.h[UPPER]; break;
	　case 2:　T +=SR31.h[LOWER]; break;
	　case 3:　T +=SR31.h[UPPER]; break;
	　}
	}
	base.uw = T;

1111	同上、但し、ベース・レジスタの更新を行わない。
	T = base.uw;
	for(t=0; t < rptamt; t++){
	　targetAdrs = T;
	　switch((ISW.h[UPPER] >> (2 * t)) & 0x3){
	　case 0:　T += index.h[LOWER]; break;
	　case 1:　T += index.h[UPPER]; break;
	　case 2:　T +=SR31.h[LOWER]; break;
	　case 3:　T +=SR31.h[UPPER]; break;
	　}
	}

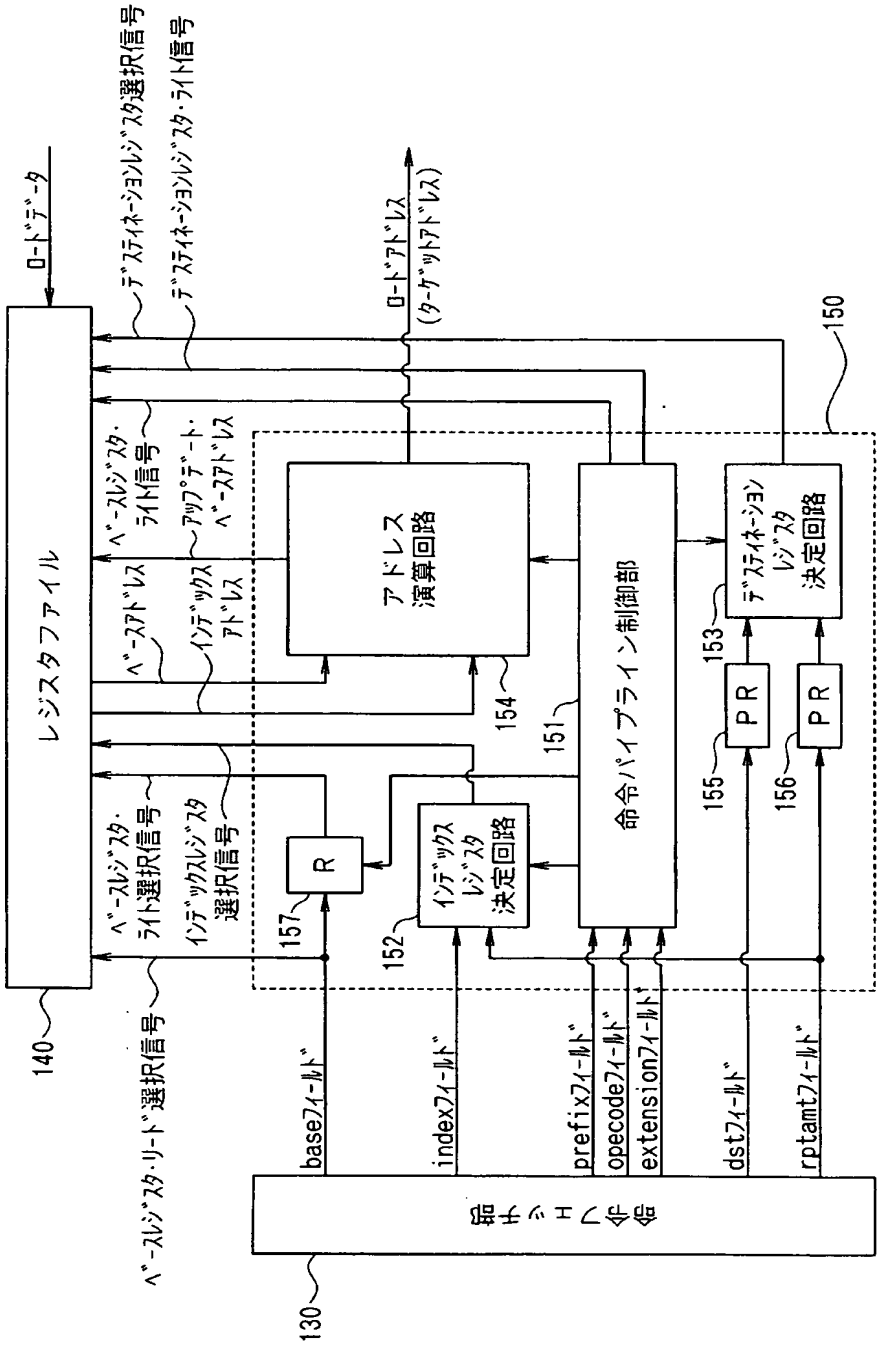
【図９】


110
メモリ
120
メモリ制御部
130
命令フェッチ部
140
レジスタファイル
150
ロード
ユニット
160
ストア
ユニット
170
演算ユニット
SR0 SR8 SR16 SR24
SR1 SR9 SR17 SR25
SR2 SR10 SR18 SR26
SR3 SR11 SR19 SR27
SR4 SR12 SR20 SR28
SR5 SR13 SR21 SR29
SR6 SR14 SR22 SR30
SR7 SR15 SR23 SR31
143
141
142
vr00 vr10 vr20 vr30
vr01 vr11 vr21 vr31
vr02 vr12 vr22 vr32
vr03 vr13 vr23 vr33
vr04 vr14 vr24 vr34
vr05 vr15 vr25 vr35
vr06 vr16 vr26 vr36
vr07 vr17 vr27 vr37
VR0 VR1 VR2 VR3
vr40 vr50 vr60 vr70
vr41 vr51 vr61 vr71
vr42 vr52 vr62 vr72
vr43 vr53 vr63 vr73
vr44 vr54 vr64 vr74
vr45 vr55 vr65 vr75
vr46 vr56 vr66 vr76
vr47 vr57 vr67 vr77
VR4 VR5 VR6 VR7
2

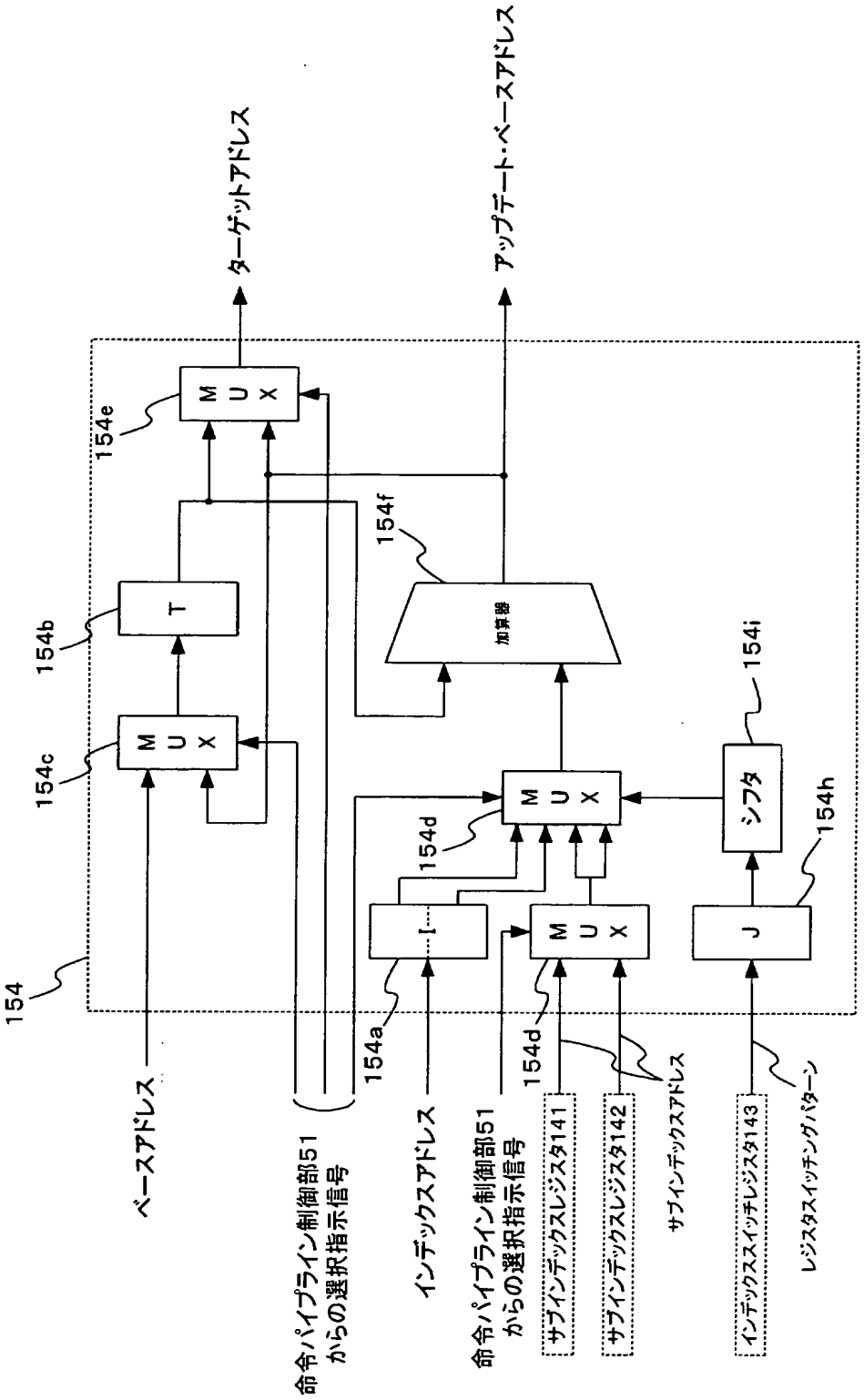
【図１０】


31
16 15
0
サブインデックスレジスタ142の場合のスイッチングパターン
サブインデックスレジスタ141の場合のスイッチングパターン
第8要素の指定
第1要素の指定
00:インデックスレジスタの下位16ビット
01:インデックスレジスタの上位16ビット
10:サブインデックスレジスタ141の下位16ビット
11:サブインデックスレジスタ141の上位16ビット

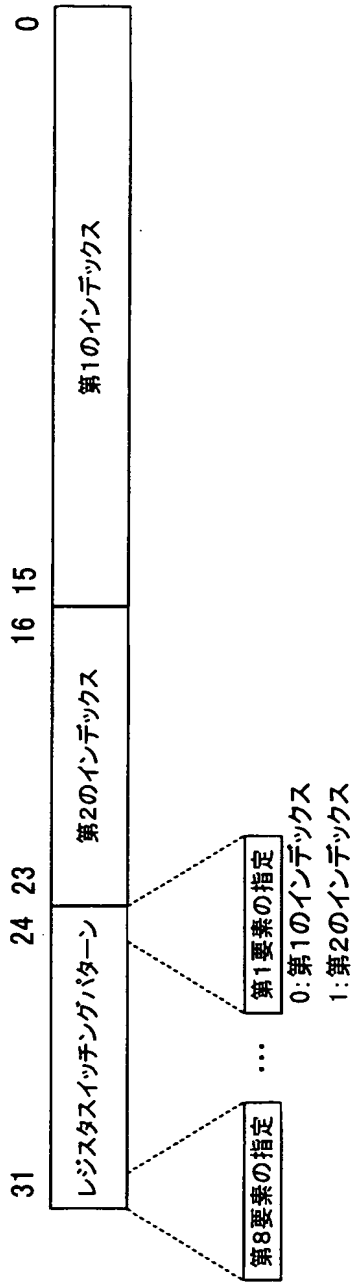
【図１１】


ロードデータ
デスティネーションレジスタ選択信号
デスティネーションレジスタ・ライト信号
ロードアドレス
(ターゲットアドレス)
150
レジスタファイル
ベースレジスタ・ライト信号
アップデートベースアドレス
アドレス演算回路
ベースアドレス
インデックスアドレス
154
151
命令パイプライン制御部
153
デスティネーションレジスタ決定回路
PR
155
156
PR
ベースレジスタ・ライト選択信号
インデックスレジスタ選択信号
R
157
152
インデックスレジスタ決定回路
140
ベースレジスタ・リード選択信号
baseフィールド
indexフィールド
prefixフィールド
opecodeフィールド
extensionフィールド
dstフィールド
rptamtフィールド
命令デコード部
130

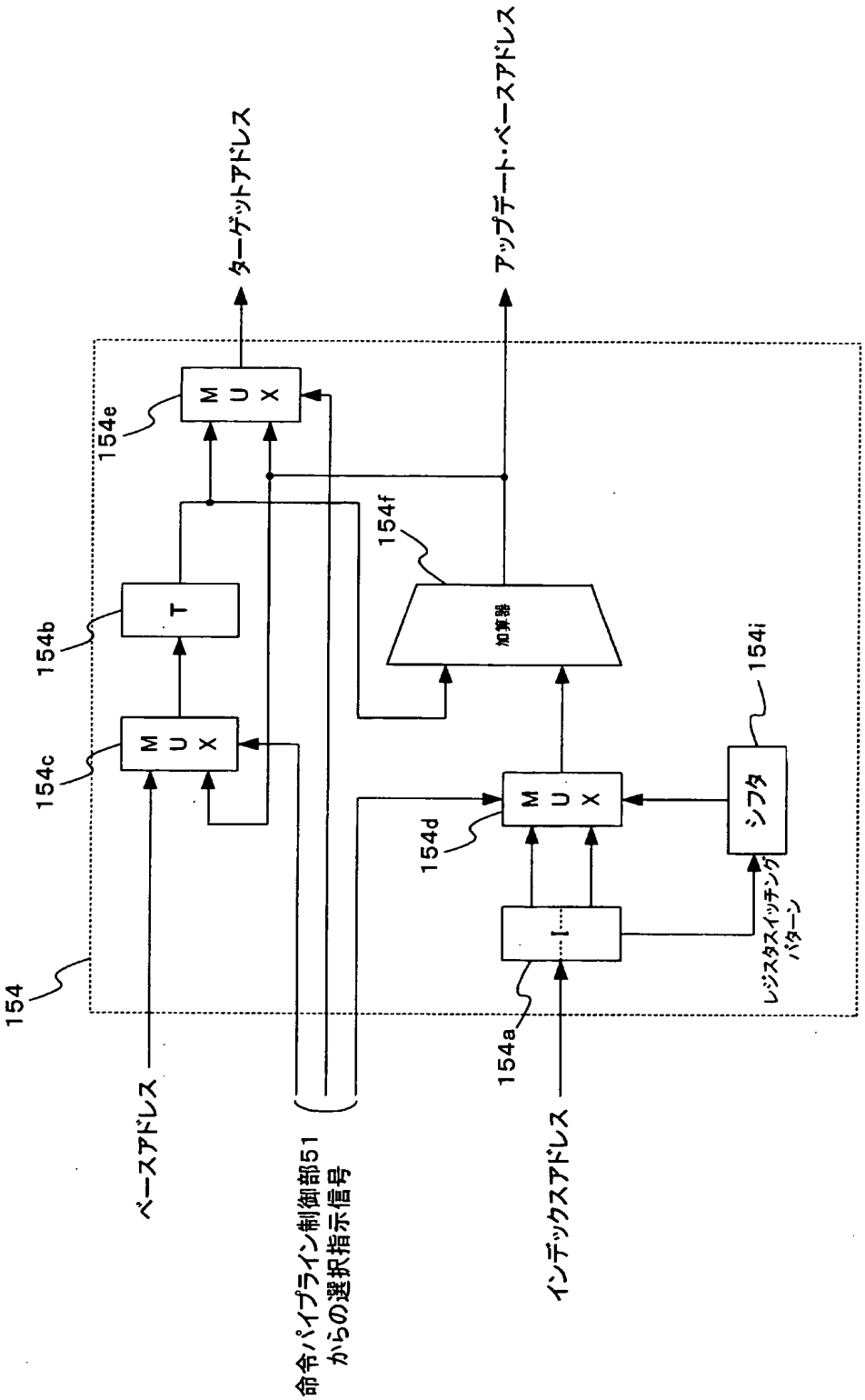
【図１２】


ターゲットアドレス
アップデート・ベースアドレス
154e
MUX
154f
加算器
154b
T
154c
MUX
154d
MUX
154i
シフタ
154h
154
I
J
154a
154d
ベースアドレス
命令パイプライン制御部51
からの選択指示信号
インデックスアドレス
命令パイプライン制御部51
からの選択指示信号
サブインデックスレジスタ141
サブインデックスレジスタ142
サブインデックスアドレス
インデックススイッチレジスタ143
レジスタスイッチングパターン

【図１３】


0
15
16
23
24
31
第1のインデックス
第2のインデックス
レジスタスイッチングパターン
第1要素の指定
第8要素の指定
0:第1のインデックス
1:第2のインデックス

【図１４】


ターゲットアドレス
アップデート・ベースアドレス
154e
MUX
154f
加算器
154b
T
154c
MUX
154d
MUX
154i
シフタ
154
154a
ベースアドレス
命令パイプライン制御部51
からの選択指示信号
インデックスアドレス
レジスタスイッチング
パターン

【書類名】　要約書

【要約】

【課題】　間接指標ベクトル参照をより効率的に行うこと。

【解決手段】　インデックスによって指定されたベクトルレジスタの要素レジスタあるいはスカラーレジスタを複数の領域に分割し、分割した領域のいずれかを選択することによって、所定のインデックスベクトル（指標ベクトル）を取得することとしている。したがって、１つのベクトルレジスタに、実質的に複数のインデックスベクトルを格納できることとなり、ベクトルレジスタを効率的に使用することが可能となる。また、インデックスベクトルを用意する手順としては、１つのインデックスベクトルの場合と同様であるため、プログラムのコードサイズや処理サイクルがほぼ増加することがない。即ち、本発明によれば、間接指標ベクトル参照をより効率的に行うことが可能となる。

【選択図】　図２

認定・付加情報

特許出願の番号	特願２００３－２０３８５６
受付番号	５０３０１２６４０４３
書類名	特許願
担当官	第七担当上席　　００９６
作成日	平成１５年　８月　４日

＜認定情報・付加情報＞

【提出日】　平成15年 7月30日

【特許出願人】

【識別番号】　000002369

【住所又は居所】　東京都新宿区西新宿２丁目４番１号

【氏名又は名称】　セイコーエプソン株式会社

【代理人】　申請人

【識別番号】　100095728

【住所又は居所】　長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社　知的財産本部内

【氏名又は名称】　上柳　雅誉

【選任した代理人】

【識別番号】　100107261

【住所又は居所】　長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社　知的財産本部内

【氏名又は名称】　須澤　修

【選任した代理人】

【識別番号】　100107076

【住所又は居所】　長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社　知的財産本部内

【氏名又は名称】　藤綱　英吉

特願２００３－２０３８５６

出　願　人　履　歴　情　報

識別番号　［０００００２３６９］

１．変更年月日　１９９０年　８月２０日
［変更理由］　新規登録
住　所　東京都新宿区西新宿２丁目４番１号
氏　名　セイコーエプソン株式会社